

```

BBBBBBBBBBBBBB      AAAAAAAAAA      SSSSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTTTT      LLL
BBBBBBBBBBBBBB      AAAAAAAAAA      SSSSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTTTT      LLL
BBBBBBBBBBBBBB      AAAAAAAAAA      SSSSSSSSSSSSSS      RRRRRRRRRRRR      TTTTTTTTTTTTTTTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAAAAAAAAAAAAAAAAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAAAAAAAAAAAAAAAAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAAAAAAAAAAAAAAAAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBB      BBB      AAA      AAA      SSS      SSS      RRR      RRR      TTT      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSSSS      RRR      RRR      TTT      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSSSS      RRR      RRR      TTT      TTT      LLL
BBBBBBBBBBBBBB      AAA      AAA      SSSSSSSSSSSSSS      RRR      RRR      TTT      TTT      LLL

```

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLL IIIII
LLLLLLLLLL IIIII
SSSSSSSS
SSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSS
SSSSSSSS

```

(2) 69
(4) 358

DECLARATIONS
BASSMAT.SUB - subtract 2 arrays giving a third


```
0000 1 .TITLE BASSMAT_SUB
0000 2 .IDENT /1-016/
0000 3
0000 4
0000 5
0000 6
0000 7
0000 8
0000 9
0000 10
0000 11
0000 12
0000 13
0000 14
0000 15
0000 16
0000 17
0000 18
0000 19
0000 20
0000 21
0000 22
0000 23
0000 24
0000 25
0000 26
0000 27
0000 28
0000 29
0000 30
0000 31
0000 32
0000 33
0000 34
0000 35
0000 36
0000 37
0000 38
0000 39
0000 40
0000 41
0000 42
0000 43
0000 44
0000 45
0000 46
0000 47
0000 48
0000 49
0000 50
0000 51
0000 52
0000 53
0000 54
0000 55
0000 56
0000 57

; File: BASMATSUB.MAR Edit: DG1016

*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****

++
FACILITY: BASIC code support

ABSTRACT:

    This module subtracts the second input array from the first input
    array and stores the result in a third array. All three arrays
    may be of any dtype except that float and double may not
    be mixed.

ENVIRONMENT: User Mode, AST Reentrant

--
AUTHOR: R. Will, CREATION DATE: 22-Jun-79

MODIFIED BY:

++
1-001 - Original
1-002 - Set IV bit in entry mask. RW 2-Oct-79
1-003 - Add support for byte, g & h floating. PLL 22-Sep-81
1-004 - Change shared external references to G^ RNH 25-Sep-81
1-005 - Substitute a macro for the calls to the array fetch and store
        routines. This should speed things up. PLL 9-Nov-81
1-006 - STORE macro must handle g & h floating. PLL 11-Nov-81
1-007 - Reserve enough space on the stack for an hfloat source. PLL 17-Nov-81
1-008 - Correct an error message.
        - Correct a run-time expression in the FETCH and STORE macros.
        PLL 20-Jan-82
1-009 - Correct FETCH, STORE again. PLL 23-Feb-82
```

```
0000 58 : 1-010 - Don't list macro expansions. PLL 16-Mar-82
0000 59 : 1-011 - Fix storing of g and h floats (use the right registers). PLL 13-Apr-82
0000 60 : 1-012 - Add support for arrays of descriptors. Also remove FETCH and
0000 61 : STORE macros - they are now in S.MLB. LEB 28-JUN-1982.
0000 62 : 1-013 - Change own storage to stack storage. LEB 9-Jul-1982
0000 63 : 1-014 - Allow gfloat result to be stored in a double destination, and
0000 64 : vice versa. PLL 7-Oct-1982
0000 65 : 1-015 - Use G* for ALL externals. SBL 16-Nov-1982
0000 66 : 1-016 - BYTE destination must use STORE_BYTE. DG 10-Jan-1984
0000 67 :--
```

DECLARATIONS

```
0000 69      .SBTTL  DECLARATIONS
0000 70      :
0000 71      : INCLUDE FILES:
0000 72      :
0000 73      :
0000 74      $DSCDEF      ; define descriptor offsets
0000 75      $SFDEF      ; use to get scale
0000 76      :
0000 77      :
0000 78      : EXTERNAL DECLARATIONS:
0000 79      :
0000 80      :
0000 81      .DSABL  GBL      ; Prevent undeclared
0000 82      :              symbols from being
0000 83      :              automatically global.
0000 84      .EXTRN  BASSK_ARGDONMAT      ; signalled if all 3 blocks
0000 85      :              not present in array desc
0000 86      :              or dimct = 0
0000 87      .EXTRN  BASSK_DATTYPERR      ; signalled if dtype of array
0000 88      :              isn't word long float double
0000 89      .EXTRN  BASSK_MATDIMERR      ; signalled if # of dims on
0000 90      :              source arrays don't agree
0000 91      .EXTRN  BASSK_ARRMUSSAM      ; signalled if upper and lower
0000 92      :              bnds not same on src arrays
0000 93      .EXTRN  BASS$TO_FA_B_R8      ; array element store for byte
0000 94      .EXTRN  BASS$TO_FA_W_R8      ; array element store for word
0000 95      .EXTRN  BASS$TO_FA_L_R8      ; array element store for long
0000 96      .EXTRN  BASS$TO_FA_F_R8      ; array element store - float
0000 97      .EXTRN  BASS$TO_FA_D_R8      ; array element store - double
0000 98      .EXTRN  BASS$TO_FA_G_R8      ; array element store - gfloat
0000 99      .EXTRN  BASS$TO_FA_H_R8      ; array element store - hfloat
0000 100     .EXTRN  BASS$FET_FA_B_R8      ; array element fetch - byte
0000 101     .EXTRN  BASS$FET_FA_W_R8      ; array element fetch - word
0000 102     .EXTRN  BASS$FET_FA_L_R8      ; array element fetch - long
0000 103     .EXTRN  BASS$FET_FA_F_R8      ; array element fetch - float
0000 104     .EXTRN  BASS$FET_FA_D_R8      ; array element fetch - double
0000 105     .EXTRN  BASS$FET_FA_G_R8      ; array element fetch - gfloat
0000 106     .EXTRN  BASS$FET_FA_H_R8      ; array element fetch - hfloat
0000 107     .EXTRN  MTH$DINT R4      ; cvt dbl to truncated dbl
0000 108     .EXTRN  BASSMAT_REDIM      ; check if redimensioning of
0000 109     :              dest array is necessary, if
0000 110     :              so, do it
0000 111     .EXTRN  BASS$$SCALE_R1      ; scale for double procision
0000 112     .EXTRN  BASS$$STOP      ; signal fatal errors
0000 113     .EXTRN  BASS$FETCH_BFA
0000 114     .EXTRN  BASS$STORE_BFA
0000 115     :
0000 116     :
0000 117     : MACROS:
0000 118     :
0000 119     :
0000 120     : $BASSMAT_SUB      subtract loop algorithm, see next page
0000 121     : FETCH          fetch an element from an array
0000 122     : STORE          store an element into an array
0000 123     :
0000 124     :
0000 125     : EQUATED SYMBOLS:
```


DECLARATIONS

```
0000 126 :
0000 127 :
00000000 0000 128 lower_bnd2 = 0 ; stack offset for temp
00000004 0000 129 lower_bnd1 = 4 ; stack offset for temp
00000008 0000 130 upper_bnd1 = 8 ; stack offset for temp
0000000C 0000 131 save_src2 = 12 ; stack offset for temp
0000001C 0000 132 value_desc = 28 ; output descriptor
0000001C 0000 133 str_len = 28 ; length field within desc
0000001E 0000 134 dtype = 30 ; data type field in desc
0000001F 0000 135 class = 31 ; class field within desc
00000020 0000 136 pointer = 32 ; pointer to data
00000024 0000 137 data = 36 ; data
00000018 0000 138 dsc$l_l1_1 = 24 ; desc offset if 1 sub
0000001C 0000 139 dsc$l_u1_1 = 28 ; desc offset if 1 sub
0000001C 0000 140 dsc$l_l1_2 = 28 ; desc offset if 2 sub
00000020 0000 141 dsc$l_u1_2 = 32 ; desc offset if 2 sub
00000024 0000 142 dsc$l_l2_2 = 36 ; desc offset if 2 sub
00000028 0000 143 dsc$l_u2_2 = 40 ; desc offset if 2 sub
0000 144 :
0000 145 :
0000 146 : OWN STORAGE:
0000 147 :
0000 148 :
0000 149 :
0000 150 :
0000 151 : PSECT DECLARATIONS:
0000 152 :
00000000 0000 153 .PSECT _BAS$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 154 EXE, RD, NOWRT, LONG
0000 155
0000 156
```

DECLARATIONS

```
0000 158 :+
0000 159 : This macro contains the looping mechanism for accessing all elements of
0000 160 : an array. It also contains all the logic for all the combinations of data
0000 161 : types and scaling. A macro is used to make it easy to maintain the parallel
0000 162 : code for all the different data types.
0000 163 :-
0000 164
0000 165 .MACRO $BASSMAT_SUB src1_dtype, src2_dtype ; subtract algorithm
0000 166
0000 167 :+
0000 168 : Loop through all the rows. Row and column upper and lower bounds have been
0000 169 : initialized on the stack.
0000 170 :-
0000 171
0000 172 LOOP_1ST SUB'src1_dtype'src2_dtype':
0000 173     MOVL     lower_bnd2(SP), R11 ; R11 has 2nd lower bound
0000 174
0000 175 :+
0000 176 : Loop through all the elements (columns) of the current row. Column lower
0000 177 : bound is initialized in R11. Column upper bound is on the stack.
0000 178 : Distinguish array by data type so that the correct fetch routine can
0000 179 : retrieve the data, the correct subtract can be done and the correct
0000 180 : store routine can be called.
0000 181 :-
0000 182
0000 183 LOOP_2ND_SUB'src1_dtype'src2_dtype':
0000 184
0000 185 :+
0000 186 : Get the data from the second source array - the subtrahend
0000 187 :-
0000 188
0000 189     MOVL     src2_matrix(AP), R0 ; pointer to 2nd src array
0000 190     MOVL     lower_bnd1(SP), R1 ; current row
0000 191     MOVL     R11, R2 ; current col
0000 192     FETCH    'src2_dtype' ; fetch data from src2 array
0000 193     MOV'src2_dtype' R0, save_src2(SP) ; store the 2nd array element
0000 194
0000 195 :+
0000 196 : Get the data from the first source array - the minuend
0000 197 :-
0000 198
0000 199     MOVL     src1_matrix(AP), R0 ; pointer to 1st src array
0000 200     MOVL     lower_bnd1(SP), R1 ; current row
0000 201     MOVL     R11, R2 ; current col
0000 202     FETCH    'src1_dtype' ; fetch data from src1 array
0000 203
0000 204 :+
0000 205 : If the data types of the 2 source arrays is the same, do the arithmetic
0000 206 : in that data type. Else convert the data to a common type and subtract.
0000 207 : If gfloat and double operands are mixed, they must be promoted to hfloat
0000 208 : for the subtract.
0000 209 :-
0000 210
0000 211     .IF      IDN      src1_dtype, src2_dtype ; src arrays are
0000 212     SUB'src1_dtype'2      save_src2(SP), R0 ; same data type
0000 213
0000 214 ; sub the source elements
```


DECLARATIONS

```
0000 215 BSBW DEST_CASE_'src1_dtype' ; go to store in dest
0000 216 .IFF ; src arrays different dtype
0000 217 .IF IDN src1_dtype, H ; source 1 is hfloat
0000 218 CVT'src2_dtype'H save_src2(SP), R4 ;
0000 219 ; cvt array2 to hfloat
0000 220 SUBH2 R4, R0 ; subtract
0000 221 BSBW DEST_CASE_H ; cvt to dest type
0000 222 .IFF ;
0000 223 .IF IDN src2_dtype, H ; source 2 is hfloat
0000 224 CVT'src1_dtype'H R0, R0 ; cvt src1 to hfloat
0000 225 SUBH2 save_src2(SP), R0 ; subtract
0000 226 BSBW DEST_CASE_H ; cvt to dest type
0000 227 .IFF ;
0000 228 .IF IDN src1_dtype, G ; source 1 is gfloat
0000 229 .IF DIF src2_dtype, D ; don't mix gfloat & dbl
0000 230 CVT'src2_dtype'G save_src2(SP), R2 ;
0000 231 ; cvt src2 to gfloat
0000 232 SUBG2 R2, R0 ; subtract
0000 233 BSBW DEST_CASE_G ; cvt to dest type
0000 234 .IFF ; gfloat & dbl
0000 235 CVTDH save_src2(SP), R4 ; promote src2 to hfloat
0000 236 CVTGH R0, R0 ; promote src1 to hfloat
0000 237 SUBH2 R4, R0 ; subtract
0000 238 BSBW DEST_CASE_H ; cvt to dest type
0000 239 .ENDC ;
0000 240 .IFF ;
0000 241 .IF IDN src2_dtype, G ; source 2 is gfloat
0000 242 .IF DIF src1_dtype, D ; & src1 not dbl
0000 243 CVT'src1_dtype'G R0, R0 ; cvt src1 to gfloat
0000 244 SUBG2 save_src2(SP), R0 ; subtract
0000 245 BSBW DEST_CASE_G ; cvt to dest type
0000 246 .IFF ; gfloat & dbl
0000 247 CVTGH save_src2(SP), R4 ; promote src2 to hfloat
0000 248 CVDH R0, R0 ; promote src1 to hfloat
0000 249 SUBH2 R4, R0 ; subtract
0000 250 BSBW DEST_CASE_H ;
0000 251 .ENDC ;
0000 252 .IFF ;
0000 253 .IF IDN src1_dtype, D ; source 1 is double
0000 254 CVT'src2_dtype'D save_src2(SP), save_src2(SP) ;
0000 255 ; cvt array2 to double & save
0000 256 MOVD R0, -(SP) ; save source1
0000 257 MOVL SF$L SAVE_FP(FP), R0 ; pass FP to get scale
0000 258 JSB G*BAS$$SCALE_R1 ; get scale in R0 & R1
0000 259 ; call a BLISS routine because
0000 260 ; the frame offsets are only
0000 261 ; defined for BLISS
0000 262 MULD2 save_src2+8(SP), R0 ; scale 2nd element (+8 becaus
0000 263 ; src1 is saved on stack)
0000 264 JSB G*MTH$DINT_R4 ; integerize
0000 265 SUBD3 R0, (SP)+, R0 ; sub 1st element & scaled 2nd
0000 266 BSBW DEST_CASE_D ; cvrt double dif to dest type
0000 267 .IFF ; 1st array not double
0000 268 .IF IDN src2_dtype, D ; is 2nd src double
0000 269 CVT'src1_dtype'D R0, -(SP) ; yes, make src1 double & save
0000 270 MOVL SF$L SAVE_FP(FP), R0 ; pass FP to get scale
0000 271 JSB G*BAS$$SCALE_R1 ; get scale in R0 & R1
```

```

; call a BLISS routine because
; the frame offsets are only
; defined for BLISS
; scale, (+8 because src2 is
; double and saved on stack
; integerize
; compute the difference
; cvrt double dif to dest type
; no double operands try float
; is 1st element float
R1 ; make 2nd element float
; subtract
; cvrt float diff to dest type
; 1st array not float
; is 2nd array float
; yes-make 1st element float
; subtract
; cvrt float diff to dest type
; no double or float, try long
; is 1st array long
R1 ; make 2nd element long
; subtract
; cnvrt long diff to dest type

; src2 is long
; cvt src1 to long
; subtract
; cnvrt long diff to dest type

; src1 is word
R1 ; cvt src2 to word
; subtract
; cvt to dest type

; src2 is word
; cvt src1 to word
; subtract
; cvt to dest type

; src1 is byte
R1 ; cvt src2 to byte
; subtract
; cvt to dest type

; cvt src1 to byte
; subtract

```

DECLARATIONS

```

0000 329      .ENDC
0000 330      .ENDC
0000 331      .ENDC
0000 332      .ENDC
0000 333
0000 334      :+
0000 335      : Have stored that element. Now see if it was the last column. If not,
0000 336      : continue with the next column. Otherwise continue to next row.
0000 337      :-
0000 338
0000 339      INCL  R11      ; get next column
0000 340      CMPL  R11, R9 ; see if last column done
0000 341      BGTR  5$
0000 342      BRW   LOOP_2ND_SUB'src1_dtype'src2_dtype' ; no, continue inner loop
0000 343
0000 344      :+
0000 345      : Have completed entire row. See if it was the last row. If not,
0000 346      : continue with next row.
0000 347      :-
0000 348
0000 349 5$:    INCL  lower_bnd1(SP) ; get next row
0000 350      CMPL  lower_bnd1(SP), upper_bnd1(SP) ; see if last row done
0000 351      BGTR  10$
0000 352      BRW   LOOP_1ST_SUB'src1_dtype'src2_dtype' ; no, continue outer loop
0000 353
0000 354 10$:   RET      ; yes, finished
0000 355
0000 356      .ENDM

```


BASSMAT_SUB - subtract 2 arrays giving

```
0000 358 .SBTTL BASSMAT_SUB - subtract 2 arrays giving a third
0000 359 :++
0000 360 : FUNCTIONAL DESCRIPTION:
0000 361 :
0000 362 : Subtract the second source array from the first source array
0000 363 : giving a third. Signal an error if the 2 arrays to be
0000 364 : subtracted do not have the same number of dimensions and the same
0000 365 : upper and lower bounds for those dimensions. Redimension the output
0000 366 : array to have the same upper bounds as the input arrays.
0000 367 : Initialize all the necessary
0000 368 : looping information on the stack. Conversions may have to be done
0000 369 : so that the sources are the same data type, so divide
0000 370 : the looping portion according to the data types. Conversion to the
0000 371 : correct destination data type will be done by a JSB to a routine,
0000 372 : instead of multiplying the number of possible combinations by 4.
0000 373 :
0000 374 : CALLING SEQUENCE:
0000 375 :
0000 376 : CALL BASSMAT_SUB (src1_array.rx.da, src2_array.rw.da, dest_matrix.wx.da)
0000 377 :
0000 378 : INPUT PARAMETERS:
0000 379 :
00000004 0000 380 : src1_matrix = 4
00000008 0000 381 : src2_matrix = 8
0000 382 :
0000 383 : IMPLICIT INPUTS:
0000 384 :
0000 385 : Scale from the callers frame to scale double precision.
0000 386 :
0000 387 : OUTPUT PARAMETERS:
0000000C 0000 388 :
0000 389 : dest_matrix = 12
0000 390 :
0000 391 : IMPLICIT OUTPUTS:
0000 392 :
0000 393 : NONE
0000 394 :
0000 395 : FUNCTION VALUE:
0000 396 : COMPLETION CODES:
0000 397 :
0000 398 : NONE
0000 399 :
0000 400 : SIDE EFFECTS:
0000 401 :
0000 402 : This routine calls the redimensioning routine and the array element
0000 403 : fetch and store routines and therefore may signal any of their errors.
0000 404 : It may also signal any of the errors listed in the externals section.
0000 405 : It may also cause the destination array to have different dimensions.
0000 406 :
0000 407 : --
0000 408 :
4FFC 0000 409 : .ENTRY BASSMAT_SUB, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,IV>
0002 410 :
0002 411 : +
0002 412 : REGISTER USAGE
0002 413 : R0 - R8 destroyed by store routines
0002 414 : R9 upper bound for 2nd subscript
```

BASSMAT_SUB - subtract 2 arrays giving

6-SEP-1984 10:31:08

[BASRTL.SRC]BASMATSUB.MAR;1

(4)

```
0002 415 : R10 pointer to dest array descriptor
0002 416 : R11 current value of 2nd subscript
0002 417 :-
0002 418
0002 419 :+
0002 420 : Put routine arguments into registers for ease of use.
0002 421 : If block 2 of array descriptor (multipliers) is not present then error.
0002 422 :-
0002 423
0002 424 MOVL src1_matrix(AP), R2 ; ptr to src1 array descr
1F 0A A2 07 E1 0006 425 BBC #DSC$V_FL_BOUNDS, DSC$B_AFLAGS(R2), ERR_ARGDONMAT ; exit if block 3 not
000B 426 ; present in descriptor
000B 427
000B 428 MOVL src2_matrix(AP), R3 ; ptr to src2 array descr
16 0A A3 07 E1 000F 429 BBC #DSC$V_FL_BOUNDS, DSC$B_AFLAGS(R3), ERR_ARGDONMAT ; exit if block 3 not
0014 430 ; present in descriptor
0014 431
0014 432 MOVL dest_matrix(AP), R10 ; ptr to dest descriptor
7E 7C 0018 433 CLRQ -(SP)
7E 7C 001A 434 CLRQ -(SP)
7E 7C 001C 435 CLRQ -(SP)
7E 7C 001E 436 CLRQ -(SP) ; reserve space to save src1
7E 7C 0020 437 CLRQ -(SP) ; src1 may be hfloat
0022 438 :+
0022 439 : Set up limits for looping through all elements
0022 440 :-
0022 441
0022 442 CMPB DSC$B_DIMCT(R2), #1 ; determine # of subscripts
01 0B A2 91 0022 443 BEQLU INIT_ONE_SUB ; 1 sub, go init
0F 13 0026 444 BGTRU INIT_TWO_SUBS ; >=2 subs, go init
59 1A 0028 445 ; 0 subs, fall into error proc
002A 446
002A 447 ERR_ARGDONMAT:
002A 448 PUSHL #BASSK_ARGDONMAT ; signal error, 0 for dimct
0030 449 CALLS #1, G^BASS$STOP ; or block 2 or 3 absent
0037 450
0037 451 :+
0037 452 : There is only 1 subscript. Redimension the destination array.
0037 453 : Make both upper and lower bound for 2nd
0037 454 : subscript a 1. A second subscript will be passed to and ignored by the
0037 455 : store routine. Put bounds for 1st subscript on stack.
0037 456 :-
0037 457
0037 458 INIT_ONE_SUB:
0B A3 01 91 0037 459 CMPB #1, DSC$B_DIMCT(R3) ; do src arrays have same
003B 460 ; number of dimensions
003B 461 BNEQU ERR_MATDIMERR ; no, error
1C A3 1C A2 91 003D 462 CMPB dsc$$_u1_1(R2), dsc$$_u1_1(R3) ; do src arrays have the same
0042 463 ; upper bounds
0042 464 BNEQU ERR_ARRMUSSAM ; no, error
18 A3 18 A2 91 0044 465 CMPB dsc$$_l1_1(R2), dsc$$_l1_1(R3) ; do src arrays have the same
0049 466 ; lower bounds
0049 467 BNEQU ERR_ARRMUSSAM ; no, error
1C A3 DD 004B 468 PUSHL dsc$$_u1_1(R3) ; get bound for redim
5A DD 004E 469 PUSHL R10 ; pointer to dest array desc
00000000'GF 02 FB 0050 470 CALLS #2, G^BASSMAT_REDIM ; redimension the dest
1C A3 DD 0057 471 PUSHL dsc$$_u1_1(R3) ; 1st upper bound
```

BASSMAT_SUB - subtract 2 arrays giving

6-SEP-1984 10:31:08

[BASRTL.SRC]BASMATSUB.MAR;1

```

18 A3 DD 005A 472      PUSHL dsc$L_L1_1(R3)      ; 1st lower bound
      03 14 005D 473      BGTR 1$                ; not 0 or neg, do 2nd sub
6E 01 DD 005F 474      MOVL #1, (SP)             ; don't alter col 0
      01 DD 0062 475 1$: PUSHL #1                ; dummy 2nd upper bound
59 01 DD 0064 476      MOVL #1, R9               ; dummy 2nd lower bound
      62 11 0067 477      BRB SEPARATE_DTYPES      ; go loop
      0069 478
      0069 479 ERR_MATDIMERR:
00000000'8F DD 0069 480      PUSHL #BASSK_MATDIMERR      ; Signal error, src arrays
00000000'GF 01 FB 006F 481      CALLS #1, G^BASS$STOP      ; don't have same # dimensns
      0076 482
      0076 483 ERR_ARRMUSSAM:
00000000'8F DD 0076 484      PUSHL #BASSK_ARRMUSSAM      ; Signal error, src arrays
00000000'GF 01 FB 007C 485      CALLS #1, G^BASS$STOP      ; same bounds
      0083 486
      0083 487 ;+
      0083 488 ; There are 2 subscripts. Check and redimension the destination array if
      0083 489 ; necessary. Put the upper bound for both subscripts on the
      0083 490 ; stack and make sure that the lower bound for both subscripts will start
      0083 491 ; at 1 (do not alter row or col 0)
      0083 492 ; -
      0083 493
      0083 494 INIT_TWO_SUBS:
      0083 495      CMPB #2, DSC$B_DIMCT(R3)          ; do src arrays have same
      0087 496                      ; number of dimensions
      0087 497      BNEQU ERR_MATDIMERR              ; no, error
20 A3 20 A2 91 0089 498      CMPB dsc$L_u1_2(R2), dsc$L_u1_2(R3) ; do src arrays have the same
      008E 499                      ; 1st upper bounds
      008E 500      BNEQU ERR_ARRMUSSAM              ; no, error
1C A3 1C A2 91 0090 501      CMPB dsc$L_L1_2(R2), dsc$L_L1_2(R3) ; do src arrays have the same
      0095 502                      ; 1st lower bounds
      0095 503      BNEQU ERR_ARRMUSSAM              ; no, error
28 A3 28 A2 91 0097 504      CMPB dsc$L_u2_2(R2), dsc$L_u2_2(R3) ; do src arrays have the same
      009C 505                      ; 2nd upper bounds
      009C 506      BNEQU ERR_ARRMUSSAM              ; no, error
24 A3 24 A2 91 009E 507      CMPB dsc$L_L2_2(R2), dsc$L_L2_2(R3) ; do src arrays have the same
      00A3 508                      ; 2nd lower bounds
      00A3 509      BNEQU ERR_ARRMUSSAM              ; no, error
      00A3 510      PUSHL dsc$L_u2_2(R3)              ; 2nd upper bound
      00A5 511      PUSHL dsc$L_u1_2(R3)              ; 1st upper bound
      00A8 512      PUSHL R10                        ; dest array pointer
00000000'GF 03 FB 00AD 513      CALLS #3, G^BASS$MAT REDIM ; redimension destination
      00AB 514      PUSHL dsc$L_u1_2(R3)              ; 1st upper bound
      00AD 515      PUSHL dsc$L_L1_2(R3)              ; 1st lower bound
      00B7 516      BGTR 1$                          ; not row 0 or neg, do cols
      00BA 517      MOVL #1, (SP)                    ; start with row 1
      00BC 518 1$: MOVL dsc$L_u2_2(R3), R9            ; 2nd upper bound
59 28 A3 DD 00BF 519      PUSHL dsc$L_L2_2(R3)        ; 2nd lower bound
      24 A3 DD 00C3 520      BGTR SEPARATE_DTYPES      ; not col 0 or neg, go loop
      03 14 00C6 521      MOVL #1, (SP)              ; start with col 1
      00C8 522
      00CB 523 ;+
      00CB 524 ; Algorithm now differs according to data types
      00CB 525 ; -
      00CB 526
      00CB 527 SEPARATE_DTYPES:
05 06 02 A2 8F 00CB 528 5$: CASEB DSC$B_DTYPE(R2), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
```


BASSMAT_SUB - subtract 2 arrays giving

```
0037' 00D0 529 2$: .WORD BYTE-2$ ; code for byte dtype
0E22' 00D2 530 .WORD WORD-2$ ; code for word dtype
1C0D' 00D4 531 .WORD LONG-2$ ; code for long dtype
002A' 00D6 532 .WORD ERR_DATTYPERR-2$ ; quad not supported
29F8' 00D8 533 .WORD FLOAT-2$ ; code for float dtype
37E3' 00DA 534 .WORD DOUBLE-2$ ; code for double dtype
      00DC 535
      00DC 536
      00DC 537 ;+ G and h floating dtype numbers fall outside the range of the CASEB, so
      00DC 538 ; check for them separately.
      00DC 539 ;+
      00DC 540
1B 02 A2 91 00DC 541 CMPB DSC$B_DTYPE(R2), #DSC$K_DTYPE_G
      03 12 00E0 542 BNEQ 3$
      460E 31 00E2 543 BRW GFLOAT
1C 02 A2 91 00E5 544 3$: CMPB DSC$B_DTYPE(R2), #DSC$K_DTYPE_H
      03 12 00E9 545 BNEQ 4$
      5414 31 00EB 546 BRW HFLOAT
18 02 A2 91 00EE 547 4$: CMPB DSC$B_DTYPE(R2), #DSC$K_DTYPE_DSC
      06 12 00F2 548 BNEQ ERR_DATTYPERR
52 04 A2 D0 00F4 549 MOVL 4(R2), R2 ; R2 <-- addr of descriptor
      D1 11 00F8 550 BRB 5$ ; CASE again on dtype in desc
      00FA 551
      00FA 552
      00FA 553
      00FA 554 ERR_DATTYPERR:
00000000'8F DD 00FA 555 PUSHL #BASS$K_DATTYPERR ; Signal error, unsupported
00000000'GF 01 FB 0100 556 CALLS #1, G*BASS$$STOP ; dtype in array desc
```

```
05 06 02 A3 8F 0107 559 ;+
      002D' 0107 560 ; Source array is a byte array. Now differentiate on the destination type.
      021B' 0107 561 ; -
      040C' 0107 562
      FFEE' 0107 563 BYTE: CASEB DSC$B_DTYPE(R3), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
      05FD' 010C 564 1$: .WORD BYTE_TO_BYTE-1$ ; code for byte dtype
      07EE' 010E 565 .WORD BYTE_TO_WORD-1$ ; code for word dtype
      0110 566 .WORD BYTE_TO_LONG-1$ ; code for long dtype
      0112 567 .WORD ERR_DATTYPERR-1$ ; quad not supported
      0114 568 .WORD BYTE_TO_FLOAT-1$ ; code for float dtype
      0116 569 .WORD BYTE_TO_DOUBLE-1$ ; code for double dtype
      0118 570
1B 02 A3 91 0118 571 CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_G
      03 12 011C 572 BNEQ 2$
      09DD 31 011E 573 BRW BYTE_TO_GFLOAT
      0121 574
1C 02 A3 91 0121 575 2$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_H
      03 12 0125 576 BNEQ 3$
      0BCE 31 0127 577 BRW BYTE_TO_HFLOAT
      012A 578
1B 02 A3 91 012A 579 3$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_DSC
      06 12 012E 580 BNEQ 4$
53 04 A3 D0 0130 581 MOVL 4(R3), R3 ; R3 <-- addr of descriptor
      D1 11 0134 582 BRB BYTE ; CASE again on dtype in desc
      0136 583
      FFC1 31 0136 584 4$: BRW ERR_DATTYPERR
      0139 585
      0139 586 ;+
      0139 587 ; Now type of source and destination arrays are known. Use the macro to
      0139 588 ; generate the code for each case
      0139 589 ; -
```

BASSMAT_SUB
1-016

E 9
BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 14
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)
0139 591 BYTE_TO_BYTE: \$BASSMAT_SUB B, B
0327 592

BASSMAT_SUB
1-016

F 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 15
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

0327 594 BYTE_TO_WORD: \$BASSMAT_SUB B, W
0518 595

BASSMAT_SUB
1-016

G 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 16
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

0518 597 BYTE_TO_LONG: \$BASSMAT_SUB B, L
0709 598

BASSMAT_SUB
1-016

H 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 17
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

0709 600 BYTE_TO_FLOAT: \$BASSMAT_SUB B. F
08FA 601

BASSMAT_SUB
1-016

I 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 18
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

08FA 603 BYTE_TO_DOUBLE: SBASSMAT_SUB B, D
0AFE 604

BASSMAT_SUB
1-016

J 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 19
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

DAFE 606 BYTE_TO_GFLOAT: SBASSMAT_SUB B, G
OCFB 607

```
OCF8 609 BYTE_TO_HFLOAT: $BASSMAT_SUB B, H
OEF2 610
OEF2 611 ;+
OEF2 612 ; Source array is a word array. Now differentiate on the destination type.
OEF2 613 ; -
OEF2 614
05 06 02 A3 8F OEF2 615 WORD: CASEB DSC$B_DTYPE(R3), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
002D OEF7 616 1$: .WORD WORD_TO_BYTE-1$ ; code for byte dtype
021E OEF9 617 .WORD WORD_TO_WORD-1$ ; code for word dtype
040C OEFB 618 .WORD WORD_TO_LONG-1$ ; code for long dtype
F203 OEFD 619 .WORD ERR_DATTYPERR-1$ ; quad not supported
05FD OEFF 620 .WORD WORD_TO_FLOAT-1$ ; code for float dtype
07EE OF01 621 .WORD WORD_TO_DOUBLE-1$ ; code for double dtype
OF03 622
1B 02 A3 91 OF03 623 CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_G
03 12 OF07 624 BNEQ 2$
09DD 31 OF09 625 BRW WORD_TO_GFLOAT
OF0C 626
1C 02 A3 91 OF0C 627 2$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_H
03 12 OF10 628 BNEQ 3$
0BCE 31 OF12 629 BRW WORD_TO_HFLOAT
OF15 630
18 02 A3 91 OF15 631 3$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_DSC
06 12 OF19 632 BNEQ 4$
53 04 A3 D0 OF1B 633 MOVL 4(R3), R3 ; R3 <-- addr of descriptor
D1 11 OF1F 634 BRB WORD ; CASE again on dtype in desc
OF21 635
F1D6 31 OF21 636 4$: BRW ERR_DATTYPERR
OF24 637
OF24 638 ;+
OF24 639 ; Now type of source and destination arrays are known. Use the macro to
OF24 640 ; generate the code for each case
OF24 641 ; -
```

BASSMAT_SUB
1-016

L 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 21
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

OF24 643 WORD_TO_BYTE: \$BASSMAT_SUB W, B
1115 644

BASSMAT_SUB
1-016

M 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 22
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

1115 646 WORD_TO_WORD: SBASSMAT_SUB W, W
1303 647

BASSMAT_SUB
1-016

N 9

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 23
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

1303 649 WORD_TO_LONG: \$BASSMAT_SUB W, L
14F4 650

BASSMAT_SUB
1-016

B 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 24
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

14F4 652 WORD_TO_FLOAT: \$BASSMAT_SUB W, F
16E5 653

BASSMAT_SUB
1-016

C 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 25
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

16E5 655 WORD_TO_DOUBLE: \$BASSMAT_SUB W, D
18E9 656

BASSMAT_SUB
1-016

D 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 26
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

18E9 658 WORD_TO_GFLOAT: \$BASSMAT_SUB W, G
1AE3 659

BASSMAT_SUB
1-016

E 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 27
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

1AE3 661 WORD_TO_HFLOAT: SBASSMAT_SUB W, H
1CDD 662

```

05 06 02 A3 8F 1CDD 664 :+
1CDD 665 : Source array is a longword array. Now differentiate on the destination type
1CDD 666 :-
1CDD 667
05 06 02 A3 8F 1CDD 668 LONG: CASEB DSC$B_DTYPE(R3), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
002D' 1CE2 669 1$: .WORD LONG_TO_BYTE-1$ ; code for byte dtype
021E' 1CE4 670 .WORD LONG_TO_WORD-1$ ; code for word dtype
040F' 1CE6 671 .WORD LONG_TO_LONG-1$ ; code for long dtype
E418' 1CE8 672 .WORD ERR_DATTYPERR-1$ ; quad not supported
05FD' 1CEA 673 .WORD LONG_TO_FLOAT-1$ ; code for float dtype
07EE' 1CEC 674 .WORD LONG_TO_DOUBLE-1$ ; code for double dtype
1CEE 675
1CEE 676
1B 02 A3 91 1CEE 677 CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_G
03 12 1CF2 678 BNEQ 2$
09DD 31 1CF4 679 BRW LONG_TO_GFLOAT
1CF7 680
1C 02 A3 91 1CF7 681 2$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_H
03 12 1CFB 682 BNEQ 3$
0BCE 31 1CFD 683 BRW LONG_TO_HFLOAT
1D00 684
1B 02 A3 91 1D00 685 3$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_DSC
06 12 1D04 686 BNEQ 4$
53 04 A3 D0 1D06 687 MOVL 4(R3), R3 ; R3 <-- addr of descriptor
D1 11 1DOA 688 BRB LONG ; CASE again on dtype in desc
1DOC 689
E3EB 31 1DOC 690 4$: BRW ERR_DATTYPERR
1DOF 691
1DOF 692 :+
1DOF 693 : Now type of source and destination arrays are known. Use the macro to
1DOF 694 : generate the code for each case
1DOF 695 :-

```

BASSMAT_SUB
1-016

G 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 29
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

1DOF 697 LONG_TO_BYTE: \$BASSMAT_SUB L, B
1F00 698

BASSMAT_SUB
1-016

H 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 30
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

1F00 700 LONG_TO_WORD: SBASSMAT_SUB L. W
20F1 701

BASSMAT_SUB
1-016

I 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 31
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

20F1 703 LONG_TO_LONG: \$BASSMAT_SUB L, L
22DF 704

BASSMAT_SUB
1-016

J 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 32
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

22DF 706 LONG_TO_FLOAT: SBASSMAT_SUB L, F
24D0 707

BASSMAT_SUB
1-016

K 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 33
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

24D0 709 LONG_TO_DOUBLE: SBASSMAT_SUB L, D
26D4 710

BASSMAT_SUB
1-016

L 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 34
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MOR;1 (5)

26D4 712 LONG_TO_GFLOAT: \$BASSMAT_SUB L, G
28CE 713

BASSMAT_SUB
1-016

M 10

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 35
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

28CE 715 LONG_TO_HFLOAT: SBASSMAT_SUB L, H
2AC8 716

BASSMAT_SUB - subtract 2 arrays giving

```
2AC8 718 ;+
2AC8 719 ; Source array is a floating array. Now differentiate on the destination type
2AC8 720 ; -
2AC8 721
05 06 02 A3 8F 2AC8 722 FLOAT: CASEB DSC$B_DTYPE(R3), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
002D' 2ACD 723 1$: .WORD FLOAT_TO_BYTE-1$ ; code for byte dtype
021E' 2ACF 724 .WORD FLOAT_TO_WORD-1$ ; code for word dtype
040F' 2AD1 725 .WORD FLOAT_TO_LONG-1$ ; code for long dtype
D62D' 2AD3 726 .WORD ERR_DATTYPERR-1$ ; quad not supported
0600' 2AD5 727 .WORD FLOAT_TO_FLOAT-1$ ; code for float dtype
07EE' 2AD7 728 .WORD FLOAT_TO_DOUBLE-1$ ; code for double dtype
2AD9 729
2AD9 730
1B 02 A3 91 2AD9 731 CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_G
03 12 2ADD 732 BNEQ 2$
09DD 31 2ADF 733 BRW FLOAT_TO_GFLOAT
2AE2 734
1C 02 A3 91 2AE2 735 2$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_H
03 12 2AE6 736 BNEQ 3$
0BCE 31 2AE8 737 BRW FLOAT_TO_HFLOAT
2AEB 738
18 02 A3 91 2AEB 739 3$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_DSC
06 12 2AEF 740 BNEQ 4$
53 04 A3 D0 2AF1 741 MOVL 4(R3), R3 ; R3 <-- addr of descriptor
D1 11 2AF5 742 BRB FLOAT ; CASE again on dtype in desc
2AF7 743
D600 31 2AF7 744 4$: BRW ERR_DATTYPERR
2AFA 745
2AFA 746 ;+
2AFA 747 ; Now type of source and destination arrays are known. Use the macro to
2AFA 748 ; generate the code for each case
2AFA 749 ; -
```

BASSMAT_SUB
1-016

B 11

BASSMAT_SUB - subtract 2 arrays giving

15-SEP-1984 23:52:49
6-SEP-1984 10:31:08

VAX/VMS Macro V04-00
[BASRTL.SRC]BASMATSUB.MAR;1

Page 37
(5)

2AFA 751 FLOAT_TO_BYTE: \$BASSMAT_SUB F, B
2CEB 752

BASSMAT_SUB
1-016

C 11

BASSMAT_SUB - subtract 2 arrays giving

15-SEP-1984 23:52:49
6-SEP-1984 10:31:08

VAX/VMS Macro V04-00
[BASRTL.SRC]BASMATSUB.MAR;1

Page 38
(5)

2CEB 754 FLOAT_TO_WORD: \$BASSMAT_SUB F, W
2EDC 755

BASSMAT_SUB
1-016

D 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 39
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

2EDC 757 FLOAT_TO_LONG: SBASSMAT_SUB F, L
30CD 758

BASSMAT_SUB
1-016

E 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 40
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

30CD 760 FLOAT_TO_FLOAT: \$BASSMAT_SUB F, F
32BB 761

BASSMAT_SUB
1-016

F 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 41
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

32BB 763 FLOAT_TO_DOUBLE: \$BASSMAT_SUB F, D
34BF 764

BASSMAT_SUB
1-016

G 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 42
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

34BF 766 FLOAT_TO_GFLOAT: SBASSMAT_SUB F, G
36B9 767

BASSMAT_SUB
1-016

H 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 43
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

3689 769 FLOAT_TO_HFLOAT: \$BASSMAT_SUB F, H
3883 770

```

38B3 772 :+
38B3 773 : Source array is a double array. Now differentiate on the destination type.
38B3 774 :-
38B3 775
05 06 02 A3 8F 38B3 776 DOUBLE: CASEB DSC$B_DTYPE(R3), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
002D 38B8 777 1$: .WORD DOUBLE_TO_BYTE-1$ : code for byte dtype
0237 38BA 778 .WORD DOUBLE_TO_WORD-1$ : code for word dtype
0441 38BC 779 .WORD DOUBLE_TO_LONG-1$ : code for long dtype
C842 38BE 780 .WORD ERR_DATTYPERR-1$ : quad not supported
0648 38C0 781 .WORD DOUBLE_TO_FLOAT-1$ : code for float dtype
0855 38C2 782 .WORD DOUBLE_TO_DOUBL-1$ : code for double dtype
38C4 783
38C4 784
1B 02 A3 91 38C4 785 CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_G
03 12 38C8 786 BNEQ 2$
0A2E 31 38CA 787 BRW DOUBLE_TO_GFLOA
38CD 788
1C 02 A3 91 38CD 789 2$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_H
03 12 38D1 790 BNEQ 3$
0C23 31 38D3 791 BRW DOUBLE_TO_HFLOA
38D6 792
1B 02 A3 91 38D6 793 3$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_DSC
06 12 38DA 794 BNEQ 4$
53 04 A3 D0 38DC 795 MOVL 4(R3), R3 : R3 <-- addr of descriptor
D1 11 38E0 796 BRB DOUBLE : CASE again on dtype in desc
38E2 797
C815 31 38E2 798 4$: BRW ERR_DATTYPERR
38E5 799
38E5 800 :+
38E5 801 : Now type of source and destination arrays are known. Use the macro to
38E5 802 : generate the code for each case
38E5 803 :-

```


BASSMAT_SUB
1-016

J 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 45
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

38E5 805 DOUBLE_TO_BYTE: SBASSMAT_SUB D, B
3AEF 806

BASSMAT_SUB
1-016

K 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 46
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

3AEF 808 DOUBLE_TO_WORD: \$BASSMAT_SUB D, W
3CF9 809

BASSMAT_SUB
1-016

L 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 47
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

3CF9 811 DOUBLE_TO_LONG: SBASSMAT_SUB D, L
3F03 812

BASSMAT_SUB
1-016

M 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 48
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

3F03 814 DOUBLE_TO_FLOAT: \$BASSMAT_SUB D, F
410D 815

BASSMAT_SUB
1-016

N 11

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 49
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

410D 817 DOUBLE_TO_DOUBL: SBASSMAT_SUB D, D
42FB 818

BASSMAT_SUB
1-016

B 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 50
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

42FB 820 DOUBLE_TO_GFLOA: \$BASSMAT_SUB D, G
44F9 821


```

      44F9 823 DOUBLE_TO_HFLOA: $BASSMAT_SUB D, H
      46F3 824
      46F3 825 ;+
      46F3 826 ; Source array is a gfloat array. Now differentiate on the destination type.
      46F3 827 ;+
      46F3 828 ;+
05 06 02 A3 8F 46F3 829 GFLOAT: CASEB DSC$B_DTYPE(R3), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
      002D 46F8 830 1$: .WORD GFLOAT_TO_BYTE-1$ ; code for byte dtype
      0226 46FA 831 .WORD GFLOAT_TO_WORD-1$ ; code for word dtype
      041F 46FC 832 .WORD GFLOAT_TO_LONG-1$ ; code for long dtype
      BA02 46FE 833 .WORD ERR_DATTYPERR-1$ ; quad not supported
      0618 4700 834 .WORD GFLOAT_TO_FLOAT-1$ ; code for float dtype
      0811 4702 835 .WORD GFLOAT_TO_DOUBL-1$ ; code for double dtype
      4704 836
1B 02 A3 91 4704 837 CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_G
      03 12 4708 838 BNEQ 2$
      09F9 31 470A 839 BRW GFLOAT_TO_GFLOA
      470D 840
1C 02 A3 91 470D 841 2$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_H
      03 12 4711 842 BNEQ 3$
      0BEC 31 4713 843 BRW GFLOAT_TO_HFLOA
      4716 844
18 02 A3 91 4716 845 3$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_DSC
      06 12 471A 846 BNEQ 4$
53 04 A3 D0 471C 847 MOVL 4(R3), R3 ; R3 <-- addr of descriptor
      D1 11 4720 848 BRB GFLOAT ; CASE again on dtype in desc
      4722 849
      B9D5 31 4722 850 4$: BRW ERR_DATTYPERR
      4725 851
      4725 852 ;+
      4725 853 ; Now type of source and destination arrays are known. Use the macro to
      4725 854 ; generate the code for each case
      4725 855 ;+
      4725 855 ;-
```

BASSMAT_SUB
1-016

D 12

BASSMAT_SUB - subtract 2 arrays giving

15-SEP-1984 23:52:49
6-SEP-1984 10:31:08

VAX/VMS Macro V04-00
[BASRTL.SRC]BASMATSUB.MAR;1

Page 52
(5)

4725 857 GFLOAT_TO_BYTE: SBASSMAT_SUB G, B
491E 858

BASSMAT_SUB
1-016

E 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 53
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

491E 860 GFLOAT_TO_WORD: SBASSMAT_SUB G, W
4817 861

BASSMAT_SUB
1-016

F 12

BASSMAT_SUB - subtract 2 arrays giving

15-SEP-1984 23:52:49

6-SEP-1984 10:31:08

VAX/VMS Macro V04-00

[BASRTL.SRC]BASMATSUB.MAR;1

Page 54
(5)

4B17 863 GFLOAT_TO_LONG: \$BASSMAT_SUB G, L
4D10 864

BASSMAT_SUB
1-016

G 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 55
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

4D10 866 GFLOAT_TO_FLOAT: \$BASSMAT_SUB G, F
4F09 867

BASSMAT_SUB
1-016

H 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 56
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

4F09 869 GFLOAT_TO_DOUBL: SBASSMAT_SUB G, D
5106 870

BASSMAT_SUB
1-016

I 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 57
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

5106 872 GFLOAT_TO_GFLOA: \$BASSMAT_SUB G, G
5302 873

```
5302 875 GFLOAT_TO_HFLOA: $BASSMAT_SUB G, H
5303 876
5304 877 :+
5305 878 : Source array is a hfloat array. Now differentiate on the destination type.
5306 879 :-
5307 880
05 D6 02 A3 8F 5307 881 HFLOAT: CASEB DSC$B_DTYPE(R3), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
002D 5307 882 1$: .WORD HFLOAT_TO_BYTE-1$ : code for byte dtype
0226 5309 883 .WORD HFLOAT_TO_WORD-1$ : code for word dtype
041F 530B 884 .WORD HFLOAT_TO_LONG-1$ : code for long dtype
ABF3 530D 885 .WORD ERR_DATTYPERR-1$ : quad not supported
0618 530F 886 .WORD HFLOAT_TO_FLOAT-1$ : code for float dtype
0811 5311 887 .WORD HFLOAT_TO_DOUBL-1$ : code for double dtype
5313 888
1B 02 A3 91 5313 889 CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_G
03 12 5317 890 BNEQ 2$
09F5 31 5319 891 BRW HFLOAT_TO_GFLOA
531C 892
1C 02 A3 91 531C 893 2$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_H
03 12 5320 894 BNEQ 3$
0BEC 31 5322 895 BRW HFLOAT_TO_HFLOA
5325 896
1B 02 A3 91 5325 897 3$: CMPB DSC$B_DTYPE(R3), #DSC$K_DTYPE_DSC
06 12 5329 898 BNEQ 4$
53 04 A3 D0 532B 899 MOVL 4(R3), R3 : R3 <-- addr of descriptor
D1 11 532F 900 BRB HFLOAT : CASE again on dtype in desc
5331 901
ABC6 31 5331 902 4$: BRW ERR_DATTYPERR
5334 903
5334 904 :+
5334 905 : Now type of source and destination arrays are known. Use the macro to
5334 906 : generate the code for each case
5334 907 :-
```

BASSMAT_SUB
1-016

K 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 59
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

5534 909 HFLOAT_TO_BYTE: \$BASSMAT_SUB H, B
572D 910

BASSMAT_SUB
1-016

L 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 60
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

572D 912 HFLOAT_TO_WORD: \$BASSMAT_SUB H, W
5926 913

BASSMAT_SUB
1-016

M 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 61
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

5926 915 HFLOAT_TO_LONG: \$BASSMAT_SUB H, L
5B1F 916

BASSMAT_SUB
1-016

N 12

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 62
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

SB1F 918 HFLOAT_TO_FLOAT: \$BASSMAT_SUB H, F
SD18 919

BASSMAT_SUB
1-016

B 13

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 63
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

5D18 921 HFLOAT_TO_DOUBL: \$BASSMAT_SUB H, 0
5F11 922

BASSMAT_SUB
1-016

C 13

BASSMAT_SUB - subtract 2 arrays giving 15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 64
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (5)

5F11 924 HFLOAT_TO_GFLOA: \$BASSMAT_SUB H, G
6111 925

D 13

15-SEP-1984 23:52:49
6-SEP-1984 10:31:08

VAX/VMS Macro V04-00
[BASRTL.SRC]BASMATSUB.MAR:1

Page 65
(5)

6111 927 HFLOAT_TO_HFLOA: SBASSMAT_SUB H, H
630D 928

```

630D 930 :+
630D 931 : Subtract has been in byte. Determine destination type to convert to dest.
630D 932 :-
630D 933
630D 934 DEST_CASE B:
05 06 56 5A D0 630D 935 MOVL R10, R6 ; save original pointer
06 02 A6 8F 6310 936 5$: CASEB DSC$B_DTYPE(R6), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
019A' 6315 937 1$: .WORD STORE_BYTE-1$ ; no conversion needed
027A' 6317 938 .WORD DEST_B_TO_W-1$ ; code for word dtype
0389' 6319 939 .WORD DEST_B_TO_L-1$ ; code for long dtype
9DE5' 631B 940 .WORD ERR_DATTYPERR-1$ ; quad not supported
0498' 631D 941 .WORD DEST_B_TO_F-1$ ; code for float dtype
05A7' 631F 942 .WORD DEST_B_TO_D-1$ ; code for double dtype
6321 943
1B 02 A6 91 6321 944 CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_G
03 12 6325 945 BNEQ 2$
0701 31 6327 946 BRW DEST_B_TO_G
632A 947
1C 02 A6 91 632A 948 2$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_H
03 12 632E 949 BNEQ 3$
0810 31 6330 950 BRW DEST_B_TO_H
6333 951
1B 02 A6 91 6333 952 3$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_DSC
06 12 6337 953 BNEQ 4$
56 04 A6 D0 6339 954 MOVL 4(R6), R6 ; R6 <-- addr of descriptor
D1 11 633D 955 BRB 5$ ; CASE again on dtype in desc
633F 956
9DB8 31 633F 957 4$: BRW ERR_DATTYPERR
6342 958
6342 959 :+
6342 960 : Subtract has been in word. Determine destination type to convert to dest.
6342 961 :-
6342 962
6342 963 DEST_CASE W:
05 06 56 5A D0 6342 964 MOVL R10, R6 ; save original pointer
06 02 A6 8F 6345 965 5$: CASEB DSC$B_DTYPE(R6), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
0136' 634A 966 1$: .WORD DEST_W_TO_B-1$ ; code for byte dtype
0274' 634C 967 .WORD STORE_WORD-1$ ; no conversion needed
0359' 634E 968 .WORD DEST_W_TO_L-1$ ; code for long dtype
9DB0' 6350 969 .WORD ERR_DATTYPERR-1$ ; quad not supported
0468' 6352 970 .WORD DEST_W_TO_F-1$ ; code for float dtype
0584' 6354 971 .WORD DEST_W_TO_D-1$ ; code for double dtype
6356 972
1B 02 A6 91 6356 973 CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_G
03 12 635A 974 BNEQ 2$
06D2 31 635C 975 BRW DEST_W_TO_G
635F 976
1C 02 A6 91 635F 977 2$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_H
03 12 6363 978 BNEQ 3$
07E1 31 6365 979 BRW DEST_W_TO_H
6368 980
1B 02 A6 91 6368 981 3$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_DSC
06 12 636C 982 BNEQ 4$
56 04 A6 D0 636E 983 MOVL 4(R6), R6 ; R6 <-- addr of descriptor
D1 11 6372 984 BRB 5$ ; CASE again on dtype in desc
6374 985
9DB3 31 6374 986 4$: BRW ERR_DATTYPERR
```

```

6377 987
6377 988 :+
6377 989 : Subtract has been in long. Determine destination type to convert to dest.
6377 990 :-
6377 991
6377 992 DEST_CASE L:
05 06 56 5A D0 6377 993      MOVL    R10, R6          ; save original pointer
      8F 637A 994 5$: CASEB    DSC$B_DTYPE(R6), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
      0106 637F 995 1$:      .WORD    DEST_L_TO_B-1$      ; code for byte dtype
      0215 6381 996      .WORD    DEST_L_TO_W-1$      ; code for word dtype
      034E 6383 997      .WORD    STORE_LONG-1$        ; no conversion needed
      9D7B 6385 998      .WORD    ERR_DATTYPERR-1$      ; quad not supported
      0438 6387 999      .WORD    DEST_L_TO_F-1$      ; code for float dtype
      0561 6389 1000     .WORD    DEST_L_TO_D-1$      ; code for double dtype
      638B 1001
1B 02 A6 91 638B 1002     CMPB     DSC$B_DTYPE(R6), #DSC$K_DTYPE_G
      03 12 638F 1003     BNEQ     2$
      06A3 31 6391 1004     BRW     DEST_L_TO_G
      6394 1005
1C 02 A6 91 6394 1006 2$:  CMPB     DSC$B_DTYPE(R6), #DSC$K_DTYPE_H
      03 12 6398 1007     BNEQ     3$
      07B2 31 639A 1008     BRW     DEST_L_TO_H
      639D 1009
1B 02 A6 91 639D 1010 3$:  CMPB     DSC$B_DTYPE(R6), #DSC$K_DTYPE_DSC
      06 12 63A1 1011     BNEQ     4$
56 04 A6 D0 63A3 1012     MOVL     4(R6), R6          ; R6 <-- addr of descriptor
      D1 11 63A7 1013     BRB     5$                ; CASE again on dtype in desc
      63A9 1014
      9D4E 31 63A9 1015 4$:  BRW     ERR_DATTYPERR
      63AC 1016
      63AC 1017 :+
      63AC 1018 : Subtract has been in float. Determine destination type to convert to dest.
      63AC 1019 :-
      63AC 1020
      63AC 1021 DEST_CASE F:
05 06 56 5A D0 63AC 1022     MOVL     R10, R6          ; save original pointer
      8F 63AF 1023 5$: CASEB     DSC$B_DTYPE(R6), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
      00D6 63B4 1024 1$:      .WORD    DEST_F_TO_B-1$      ; code for byte dtype
      01E5 63B6 1025      .WORD    DEST_F_TO_W-1$      ; code for word dtype
      02F4 63B8 1026      .WORD    DEST_F_TO_L-1$      ; code for long dtype
      9D46 63BA 1027      .WORD    ERR_DATTYPERR-1$      ; quad not supported
      0428 63BC 1028      .WORD    STORE_FLOAT-1$        ; no conversion needed
      053E 63BE 1029      .WORD    DEST_F_TO_D-1$      ; code for double dtype
      63C0 1030
1B 02 A6 91 63C0 1031     CMPB     DSC$B_DTYPE(R6), #DSC$K_DTYPE_G
      03 12 63C4 1032     BNEQ     2$
      0674 31 63C6 1033     BRW     DEST_F_TO_G
      63C9 1034
1C 02 A6 91 63C9 1035 2$:  CMPB     DSC$B_DTYPE(R6), #DSC$K_DTYPE_H
      03 12 63CD 1036     BNEQ     3$
      0783 31 63CF 1037     BRW     DEST_F_TO_H
      63D2 1038
1B 02 A6 91 63D2 1039 3$:  CMPB     DSC$B_DTYPE(R6), #DSC$K_DTYPE_DSC
      06 12 63D6 1040     BNEQ     4$
56 04 A6 D0 63D8 1041     MOVL     4(R6), R6          ; R6 <-- addr of descriptor
      D1 11 63DC 1042     BRB     5$                ; CASE again on dtype in desc
      63DE 1043
```

```
          9D19 31 63DE 1044 4$: BRW ERR_DATTYPERR
          63E1 1045
          63E1 1046
          63E1 1047 :+
          63E1 1048 : Subtract has been in double. Determine destination type to convert to dest.
          63E1 1049 :-
          63E1 1050
          05 06 56 5A DO 63E1 1051 DEST_CASE_D:
          02 A6 8F 63E4 1052 5$: MOVL R10, R6 ; save original pointer
          00A6: 63E9 1053 1$: CASEB DSC$B_DTYPE(R6), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
          01B5: 63EB 1054 : .WORD DEST_D_TO_B-1$ ; code for byte dtype
          02C4: 63ED 1055 : .WORD DEST_D_TO_W-1$ ; code for word dtype
          9D11: 63EF 1056 : .WORD DEST_D_TO_L-1$ ; code for long dtype
          03D3: 63F1 1057 : .WORD ERR_DATTYPERR-1$ ; quad not supported
          0562: 63F3 1058 : .WORD DEST_D_TO_F-1$ ; code for float dtype
          63F5 1059 : .WORD STORE_DOUBLE-1$ ; no conversion needed
          1B 02 A6 91 63F5 1060 CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_G
          03 12 63F9 1061 BNEQ 2$
          050C 31 63FB 1062 BRW DEST_G_TO_D
          63FE 1063
          1C 02 A6 91 63FE 1064 2$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_H
          03 12 6402 1065 BNEQ 3$
          0754 31 6404 1066 BRW DEST_D_TO_H
          6407 1067
          1B 02 A6 91 6407 1068 3$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_DSC
          06 12 640B 1069 BNEQ 4$
          56 04 A6 DO 640D 1070 MOVL 4(R6), R6 ; R6 <-- addr of descriptor
          D1 11 6411 1071 BRB 5$ ; CASE again on dtype in desc
          6413 1072
          9CE4 31 6413 1073 4$: BRW ERR_DATTYPERR
          6416 1074
          6416 1075 :+
          6416 1076 : Subtract has been in gfloat. Determine destination type to convert to dest.
          6416 1077 :-
          6416 1078
          05 06 56 5A DO 6416 1079 DEST_CASE_G:
          02 A6 8F 6419 1081 5$: MOVL R10, R6 ; save original pointer
          0087: 641E 1082 1$: CASEB DSC$B_DTYPE(R6), #DSC$K_DTYPE_B, #<DSC$K_DTYPE_D - DSC$K_DTYPE_B>
          0196: 6420 1083 : .WORD DEST_G_TO_B-1$ ; code for byte dtype
          02A5: 6422 1084 : .WORD DEST_G_TO_W-1$ ; code for word dtype
          9CDC: 6424 1085 : .WORD DEST_G_TO_L-1$ ; code for long dtype
          03B4: 6426 1086 : .WORD ERR_DATTYPERR-1$ ; quad not supported
          04EC: 6428 1087 : .WORD DEST_G_TO_F-1$ ; code for float dtype
          642A 1088 : .WORD DEST_G_TO_D-1$ ; code for double dtype
          1B 02 A6 91 642A 1089 CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_G
          03 12 642E 1090 BNEQ 2$
          062B 31 6430 1091 BRW STORE_GFLOAT
          6433 1092
          1C 02 A6 91 6433 1093 2$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_H
          03 12 6437 1094 BNEQ 3$
          0736 31 6439 1095 BRW DEST_G_TO_H
          643C 1096
          1B 02 A6 91 643C 1097 3$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_DSC
          06 12 6440 1098 BNEQ 4$
          56 04 A6 DO 6442 1099 MOVL 4(R6), R6 ; R6 <-- addr of descriptor
          D1 11 6446 1100 BRB 5$ ; CASE again on dtype in desc
```



```
BASSMAT_SUB - subtract 2 arrays giving

          9CAF 31 6448 1101
          6448 1102 4$: BRW ERR_DATTYPERR
          6448 1103
          6448 1104 ;+
          6448 1105 ; Subtract has been in hfloat. Determine destination type to convert to dest.
          6448 1106 ; -
          6448 1107
          6448 1108 DEST_CASE H:
05 06 56 5A DO 6448 1109 MOVL R10, R6 ; save original pointer
          02 A6 8F 644E 1110 5$: CASEB DSC$B_DTYPE(R6), #DSC$K_DTYPE_B, #DSC$K_DTYPE_D - DSC$K_DTYPE_B>
          0058' 6453 1111 1$: .WORD DEST_H_TO_B-1$ ; code for byte dtype
          0167' 6455 1112 .WORD DEST_H_TO_W-1$ ; code for word dtype
          0276' 6457 1113 .WORD DEST_H_TO_L-1$ ; code for long dtype
          9CA7' 6459 1114 .WORD ERR_DATTYPERR-1$ ; quad not supported
          0385' 645B 1115 .WORD DEST_H_TO_F-1$ ; code for float dtype
          04E1' 645D 1116 .WORD DEST_H_TO_D-1$ ; code for double dtype
          645F 1117
1B 02 A6 91 645F 1118 CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_G
          03 12 6463 1119 BNEQ 2$
          05F2 31 6465 1120 BRW DEST_H_TO_G
          6468 1121
1C 02 A6 91 6468 1122 2$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_H
          03 12 646C 1123 BNEQ 3$
          0705 31 646E 1124 BRW STORE_HFLOAT
          6471 1125
18 02 A6 91 6471 1126 3$: CMPB DSC$B_DTYPE(R6), #DSC$K_DTYPE_DSC
          06 12 6475 1127 BNEQ 4$
56 04 A6 DO 6477 1128 MOVL 4(R6), R6 ; R6 <-- addr of descriptor
          D1 11 647B 1129 BRB 5$ ; CASE again on dtype in desc
          647D 1130
          9C7A 31 647D 1131 4$: BRW ERR_DATTYPERR
          6480 1132
          6480 1133 DEST_W_TO_B:
50 50 33 6480 1134 CVTWB RO, RO ; convert
          2A 11 6483 1135 BRB STORE_BYTE ; go store
          6485 1136
          6485 1137 DEST_L_TO_B:
50 50 F6 6485 1138 CVTLB RO, RO ; convert
          25 11 6488 1139 BRB STORE_BYTE ; go store
          648A 1140
          648A 1141 DEST_F_TO_B:
50 50 48 648A 1142 CVTFB RO, RO ; convert
          20 11 648D 1143 BRB STORE_BYTE ; go store
          648F 1144
          648F 1145 DEST_D_TO_B:
          7E 50 70 648F 1146 MOVD RO, -(SP) ; save double
50 0C AD DO 6492 1147 MOVL SF$L_SAVE_FP(FP), RO ; pass FP to get scale
          00000000' GF 16 6496 1148 JSB G*BA$SCALE R1 ; get scale in R0 & R1
50 8E 50 67 649C 1149 DIVD3 RO, (SP)+, RO ; descale for dest
          50 50 68 64A0 1150 CVTDB RO, RO ; convert to byte
          0A 11 64A3 1151 BRB STORE_BYTE ; go store
          64A5 1152
          64A5 1153 DEST_G_TO_B:
50 50 48FD 64A5 1154 CVTGB RO, RO ; convert
          04 11 64A9 1155 BRB STORE_BYTE ; go store
          64AB 1156
          64AB 1157 DEST_H_TO_B:
```

```
BAS$MAT_SUB - subtract 2 arrays giving

50 50 68FD 64AB 1158          CVTHB  RO, RO          ; convert
64AF 1159          ; fall into store
64AF 1160 STORE_BYTE:
51 5A  D0 64AF 1161          MOVL  R10, R1          ; pointer to dest descriptor
52 08 AE  D0 64B2 1162          MOVL  lower_bnd1+4(SP), R2      ; current row (extra longword
64B6 1163          ; on top of stack for jsb)
53 5B  D0 64B6 1164          MOVL  R11, R3          ; current column
28 AE 50 90 64B9 1165          MOVB  RO, DATA+4(SP)
64BD 1166 ;+
64BD 1167 ;: Redefine the following offsets for the call to the STORE macro. The
64BD 1168 ;: BSBW to here added 4 to the stack.
64BD 1169 ;:-
64BD 1170 ;:-
00000020 64BD 1171 value_desc = 32
00000020 64BD 1172 str_len = 32
00000022 64BD 1173 dtype = 34
00000023 64BD 1174 class = 35
00000024 64BD 1175 pointer = 36
00000028 64BD 1176 data = 40
64BD 1177
64BD 1178          STORE  B          ; store
658E 1179 ;+
658E 1180 ;: Restore the following offsets.
658E 1181 ;:-
658E 1182 ;:-
0000001C 658E 1183 value_desc = 28
0000001C 658E 1184 str_len = 28
0000001E 658E 1185 dtype = 30
0000001F 658E 1186 class = 31
00000020 658E 1187 pointer = 32
00000024 658E 1188 data = 36
658E 1189
05 658E 1190          RSB          ; go continue loop
658F 1191
50 50 99 658F 1192 DEST_B_TO W:
2A 11 658F 1193          CVTBW  RO, RO          ; convert
6592 1194          BRB  STORE_WORD          ; go store
6594 1195
50 50 F7 6594 1196 DEST_L_TO W:
25 11 6594 1197          CVTLW  RO, RO          ; convert
6597 1198          BRB  STORE_WORD          ; go store
6599 1199
50 50 49 6599 1200 DEST_F_TO W:
20 11 6599 1201          CVTFW  RO, RO          ; convert
659C 1202          BRB  STORE_WORD          ; go store
659E 1203
659E 1204 DEST_D_TO W:
7E 50 70 659E 1205          MOVD  RO, -(SP)          ; save double
50 0C AD D0 65A1 1206          MOVL  SF$L SAVE FP(FP), RO      ; pass FP to get scale
00000000 GF 16 65A5 1207          JSB  G^BAS$SCALE R1          ; get scale in R0 & R1
50 8E 50 67 65AB 1208          DIVD3 RO, (SP)+, RO      ; descale for dest
50 50 69 65AF 1209          CVTDW  RO, RO          ; convert to word
0A 11 65B2 1210          BRB  STORE_WORD          ; go store
65B4 1211
50 50 49FD 65B4 1212 DEST_G_TO W:
04 11 65B4 1213          CVTDW  RO, RO          ; convert
65B8 1214          BRB  STORE_WORD          ; go store
```

```
BASSMAT_SUB - subtract 2 arrays giving

50 50 69FD 65BA 1215
65BA 1216 DEST_H_TO_W:
65BA 1217 CDTBW RO, RO ; convert
65BE 1218 ; fall into store
65BE 1219 STORE_WORD:
65BE 1220 MOVL R10, R1 ; pointer to dest descriptor
52 51 5A D0 65C1 1221 MOVL lower_bnd1+4(SP), R2 ; current row (extra longword)
08 AE D0 65C5 1222 ; on top of stack for jsb)
65C5 1223 MOVL R11, R3 ; current column
28 AE 50 B0 65C8 1224 MOVW RO, DATA+4(SP)
65CC 1225 ;+
65CC 1226 ; Redefine the following offsets for the call to the STORE macro. The
65CC 1227 ; BSBW to here added 4 to the stack.
65CC 1228 ;+
65CC 1229 ;+
00000020 65CC 1230 value_desc = 32
00000020 65CC 1231 str_len = 32
00000022 65CC 1232 dtype = 34
00000023 65CC 1233 class = 35
00000024 65CC 1234 pointer = 36
00000028 65CC 1235 data = 40
65CC 1236
65CC 1237 STORE W ; store
669D 1238 ;+
669D 1239 ; Restore the following offsets.
669D 1240 ;+
669D 1241 ;+
0000001C 669D 1242 value_desc = 28
0000001C 669D 1243 str_len = 28
0000001E 669D 1244 dtype = 30
0000001F 669D 1245 class = 31
00000020 669D 1246 pointer = 32
00000024 669D 1247 data = 36
669D 1248
05 669D 1249 RSB ; go continue loop
669E 1250
50 50 98 669E 1251 DEST_B_TO_L:
2A 11 669E 1252 CDTBL RO, RO ; convert
66A1 1253 BRB STORE_LONG ; go store
66A3 1254
50 50 32 66A3 1255 DEST_W_TO_L:
25 11 66A3 1256 CDTWL RO, RO ; convert
66A6 1257 BRB STORE_LONG ; go store
66A8 1258
50 50 4A 66A8 1259 DEST_F_TO_L:
20 11 66A8 1260 CDTFL RO, RO ; convert
66AB 1261 BRB STORE_LONG ; go store
66AD 1262
66AD 1263 DEST_D_TO_L:
7E 50 70 66AD 1264 MOVD RO, -(SP) ; save double
50 0C AD D0 66B0 1265 MOVL SF$L SAVE FP(FP), RO ; pass FP to get scale
00000000 GF 16 66B4 1266 JSB G^BASS$SCALE R1 ; get scale in R0 & R1
50 8E 50 67 66BA 1267 DIVD3 RO, (SP)+, RO ; descale for dest
50 50 6A 66BE 1268 CVDL RO, RO ; convert
0A 11 66C1 1269 BRB STORE_LONG ; go store
66C3 1270
66C3 1271 DEST_G_TO_L:
```



```
BASSMAT_SUB - subtract 2 arrays giving

50 50 4AFD 66C3 1272      CVTGL  RO, RO      ; convert
    04 11 66C7 1273      BRB    STORE_LONG  ; go store
    66C9 1274
    66C9 1275 DEST_H_TO_L:
50 50 6AFD 66C9 1276      CVTHL  RO, RO      ; convert
    66CD 1277              ; fall into store
    66CD 1278
    66CD 1279 STORE_LONG:
51 5A  D0 66CD 1280      MOVL    R10, R1      ; pointer to dest descriptor
52 0B AE  D0 66D0 1281      MOVL    lower_bnd1+4(SP), R2 ; current row (extra longword
    66D4 1282              ; on stack for jsb)
53 5B  D0 66D4 1283      MOVL    R11, R3      ; current column
28 AE 50  D0 66D7 1284      MOVL    RO, DATA+4(SP)
    66DB 1285 ;+
    66DB 1286 ; Redefine the following offsets for the call to the STORE macro. The
    66DB 1287 ; BSBW to here added 4 to the stack.
    66DB 1288 ; -
    66DB 1289
00000020 66DB 1290 value_desc = 32
00000020 66DB 1291 str_len = 32
00000022 66DB 1292 dtype = 34
00000023 66DB 1293 class = 35
00000024 66DB 1294 pointer = 36
00000028 66DB 1295 data = 40
    66DB 1296
    66DB 1297      STORE  L      ; store
    67AC 1298 ;+
    67AC 1299 ; Restore the following offsets.
    67AC 1300 ; -
    67AC 1301
0000001C 67AC 1302 value_desc = 28
0000001C 67AC 1303 str_len = 28
0000001E 67AC 1304 dtype = 30
0000001F 67AC 1305 class = 31
00000020 67AC 1306 pointer = 32
00000024 67AC 1307 data = 36
    67AC 1308
    05 67AC 1309      RSB      ; go continue loop
    67AD 1310
    67AD 1311 DEST_B_TO_F:
50 50 4C 67AD 1312      CVTBF  RO, RO      ; convert
    2A 11 67B0 1313      BRB    STORE_FLOAT  ; go store
    67B2 1314
    67B2 1315 DEST_W_TO_F:
50 50 4D 67B2 1316      CVTWF  RO, RO      ; convert
    25 11 67B5 1317      BRB    STORE_FLOAT  ; go store
    67B7 1318
    67B7 1319 DEST_L_TO_F:
50 50 4E 67B7 1320      CVTLF  RO, RO      ; convert
    20 11 67BA 1321      BRB    STORE_FLOAT  ; go store
    67BC 1322
    67BC 1323 DEST_D_TO_F:
    7E 50 70 67BC 1324      MOVD    RO, -(SP)      ; save double
50 0C AD D0 67BF 1325      MOVL    SF$L SAVE_FP(FP), RO ; pass FP to get scale
00000000 GF 16 67C3 1326      JSB    G*BAS$SCALE R1 ; get scale in RO & R1
50 8E 50 67 67C9 1327      DIVD3   RO, (SP)+, RO ; descale
    50 50 76 67CD 1328      CVTDF  RO, RO      ; convert
```

```
BASSMAT_SUB - subtract 2 arrays giving

    OA 11 67D0 1329 BRB STORE_FLOAT ; go store
        67D2 1330
    50 50 33FD 67D2 1331 DEST_G_TO_F:
        04 11 67D2 1332 CDTGF RO, RO ; convert
        67D6 1333 BRB STORE_FLOAT ; go store
        67D8 1334
    50 50 F6FD 67D8 1335 DEST_H_TO_F:
        67D8 1336 CDTHF RO, RO ; convert
        67DC 1337 ; fall into store
        67DC 1338
        67DC 1339 STORE_FLOAT:
    51 5A D0 67DC 1340 MOVL R10, R1 ; pointer to dest descriptor
    52 08 AE D0 67DF 1341 MOVL lower_bnd1+4(SP), R2 ; current row (extra longword)
        67E3 1342 ; on stack for jsb)
    53 5B D0 67E3 1343 MOVL R11, R3 ; current column
    28 AE 50 50 67E6 1344 MOVF RO, DATA+4(SP)
        67EA 1345 ;+
        67EA 1346 ; Redefine the following offsets for the call to the STORE macro. The
        67EA 1347 ; BSBW to here added 4 to the stack.
        67EA 1348 ; -
        67EA 1349
    00000020 67EA 1350 value_desc = 32
    00000020 67EA 1351 str_len = 32
    00000022 67EA 1352 dtype = 34
    00000023 67EA 1353 class = 35
    00000024 67EA 1354 pointer = 36
    00000028 67EA 1355 data = 40
        67EA 1356
        67EA 1357 STORE F ; store
        688B 1358 ;+
        688B 1359 ; Restore the following offsets.
        688B 1360 ; -
        688B 1361
    0000001C 688B 1362 value_desc = 28
    0000001C 688B 1363 str_len = 28
    0000001E 688B 1364 dtype = 30
    0000001F 688B 1365 class = 31
    00000020 688B 1366 pointer = 32
    00000024 688B 1367 data = 36
        688B 1368
    05 688B 1369 RSB ; go continue loop
        688C 1370
        688C 1371 DEST_B_TO_D:
    7E 50 6C 688C 1372 CDTBD RO, -(SP) ; save double
    50 0C AD D0 688F 1373 MOVL SF$L SAVE FP(FP), RO ; pass FP to get scale
    00000000 GF 16 68C3 1374 JSB G^BAS$$SCALE_R1 ; get scale in R0 & R1
    50 8E 64 68C9 1375 MUL2 (SP)+, RO ; scale for dest
        7D 11 68CC 1376 BRB STORE_DOUBLE ; go store
        68CE 1377
        68CE 1378 DEST_W_TO_D:
    7E 50 6D 68CE 1379 CDTWD RO, -(SP) ; save double
    50 0C AD D0 68D1 1380 MOVL SF$L SAVE FP(FP), RO ; pass FP to get scale
    00000000 GF 16 68D5 1381 JSB G^BAS$$SCALE_R1 ; get scale in R0 & R1
    50 8E 64 68DB 1382 MUL2 (SP)+, RO ; scale for dest
        68 11 68DE 1383 BRB STORE_DOUBLE ; go store
        68E0 1384
        68E0 1385 DEST_L_TO_D:
```

BASSMAT_SUB - subtract 2 arrays giving

```

      7E  50  6E  68E0 1386      CMTLD  R0, -(SP)          ; save double
50  7E  50  68E3 1387      MOVL  SF$SAVE_FP(FP), R0      ; pass FP to get scale
00000000'GF 16  68E7 1388      JSB  G^BASS$SCALE_R1       ; get scale in R0 & R1
50  8E  64  68ED 1389      MUL2  (SP)+, R0              ; scale for dest
59  11  68F0 1390      BRB  STORE_DOUBLE                ; go store
      68F2 1391
      68F2 1392      DEST_F_TO D:
50  7E  50  68F2 1393      CMTFD  R0, -(SP)          ; save double
50  7E  50  68F5 1394      MOVL  SF$SAVE_FP(FP), R0      ; pass FP to get scale
00000000'GF 16  68F9 1395      JSB  G^BASS$SCALE_R1       ; get scale in R0 & R1
50  8E  64  68FF 1396      MUL2  (SP)+, R0              ; scale for dest
00000000'GF 16  6902 1397      JSB  G^MTH$DINT_R4         ; integerize
41  11  6908 1398      BRB  STORE_DOUBLE                ; go store
      690A 1399
      690A 1400      DEST_G_TO D:
      690A 1401      ; Note the intermediate conversion to hfloat.
      690A 1402
      690A 1403
      690A 1404      MOVL  R2, -(SP)          ; save regs which CVTGH
7E  52  DO  690A 1405      MOVL  R3, -(SP)          ; will destroy
7E  53  DO  690D 1406      CVTGH  R0, R0             ; cvt gfloat to hfloat
50  50  56FD 6910 1407      CVTHD  R0, -(SP)         ; cvt to desired double
7E  50  F7FD 6914 1408      MOVL  (SP)+, R3          ; restore regs
53  8E  DO  6918 1409      MOVL  (SP)+, R2
52  8E  DO  691B 1410      MOVL  SF$SAVE_FP(FP), R0      ; pass FP to get scale
50  7E  50  6922 1411      MUL2  (SP)+, R0              ; scale
50  8E  64  6925 1412      MOVL  R4, -(SP)          ; save R4
00000000'GF 16  6928 1413      JSB  G^MTH$DINT_R4         ; integerize
54  8E  DO  692E 1414      MOVL  (SP)+, R4          ; restore R4
      0017 31  6931 1415      BRW  STORE_DOUBLE
      6934 1416
      6934 1417      DEST_H_TO D:
50  7E  50  F7FD 6934 1418      CVTHD  R0, -(SP)         ; save double
50  7E  50  6938 1419      MOVL  SF$SAVE_FP(FP), R0      ; pass FP to get scale
00000000'GF 16  693C 1420      JSB  G^BASS$SCALE_R1       ; get scale in R0 & R1
50  8E  64  6942 1421      MUL2  (SP)+, R0              ; scale for dest
00000000'GF 16  6945 1422      JSB  G^MTH$DINT_R4         ; integerize
      694B 1423      ; fall into store
      694B 1424      STORE_DOUBLE:
52  5A  DO  694B 1425      MOVL  R10, R2              ; pointer to dest descriptor
53  08  AE  DO  694E 1426      MOVL  lower_bnd1+4(SP), R3 ; current row (extra longword
      6952 1427      ; on stack for jsb)
54  5B  DO  6952 1428      MOVL  R11, R4              ; current column
28  AE  50  70  6955 1429      MOVD  R0, DATA+4(SP)
      6959 1430      ;+
      6959 1431      ; Redefine the following offsets for the call to the STORE macro. The
      6959 1432      ; BSBW to here added 4 to the stack.
      6959 1433      ; -
      6959 1434
00000020 6959 1435      value_desc = 32
00000020 6959 1436      str_len = 32
00000022 6959 1437      dtype = 34
00000023 6959 1438      class = 35
00000024 6959 1439      pointer = 36
00000028 6959 1440      data = 40
      6959 1441
      6959 1442      STORE  D          ; store
```

```
6A2A 1443 :+
6A2A 1444 : Restore the following offsets.
6A2A 1445 :-
6A2A 1446
0000001C 6A2A 1447 value_desc = 28
0000001C 6A2A 1448 str_len = 28
0000001E 6A2A 1449 dtype = 30
0000001F 6A2A 1450 class = 31
00000020 6A2A 1451 pointer = 32
00000024 6A2A 1452 data = 36
6A2A 1453
05 6A2A 1454 RSB ; go continue loop
6A2B 1455
6A2B 1456 DEST_B_TO_G:
50 50 4CFD 6A2B 1457 CDTBG RO, RO ; convert
2D 11 6A2F 1458 BRB STORE_GFLOAT ; go store
6A31 1459
6A31 1460 DEST_W_TO_G:
50 50 4DFD 6A31 1461 CDTWG RO, RO ; convert
27 11 6A35 1462 BRB STORE_GFLOAT ; go store
6A37 1463
6A37 1464 DEST_L_TO_G:
50 50 4EFD 6A37 1465 CDTLG RO, RO ; convert
21 11 6A3B 1466 BRB STORE_GFLOAT ; go store
6A3D 1467
6A3D 1468 DEST_F_TO_G:
50 50 99FD 6A3D 1469 CDTFG RO, RO ; convert
1B 11 6A41 1470 BRB STORE_GFLOAT ; go store
6A43 1471
6A43 1472 DEST_D_TO_G:
6A43 1473 :+
6A43 1474 : Note the intermediate conversion to hfloat.
6A43 1475 :-
7E 52 DO 6A43 1476 MOVL R2, -(SP) ; save regs which CVDH
7E 53 DO 6A46 1477 MOVL R3, -(SP) ; will destroy
50 50 32FD 6A49 1478 CVDH RO, RO ; cvt dbl to hfloat
50 50 76FD 6A4D 1479 CVTHG RO, RO ; cvt to desired gfloat
53 8E DO 6A51 1480 MOVL (SP)+, R3 ; restore regs
52 8E DO 6A54 1481 MOVL (SP)+, R2
0004 31 6A57 1482 BRW STORE_GFLOAT
6A5A 1483
6A5A 1484 DEST_H_TO_G:
50 50 76FD 6A5A 1485 CVTHG RO, RO ; convert
6A5E 1486 ; fall into store
6A5E 1487 STORE_GFLOAT:
52 5A DO 6A5E 1488 MOVL R10, R2 ; pointer to dest descriptor
53 08 AE DO 6A61 1489 MOVL lower_bnd1+4(SP), R3 ; current row (extra longword
6A65 1490 ; on top of stack for jsb)
54 5B DO 6A65 1491 MOVL R11, R4 ; current column
28 AE 50 50FD 6A68 1492 MOVG RO, DATA+4(SP)
6A6D 1493 :+
6A6D 1494 : Redefine the following offsets for the call to the STORE macro. The
6A6D 1495 : BSBW to here added 4 to the stack.
6A6D 1496 :-
6A6D 1497
00000020 6A6D 1498 value_desc = 32
00000020 6A6D 1499 str_len = 32
```


BASSMAT_SUB - subtract 2 arrays giving

```
00000022 6A6D 1500 dtype = 34
00000023 6A6D 1501 class = 35
00000024 6A6D 1502 pointer = 36
00000028 6A6D 1503 data = 40
        6A6D 1504
        6A6D 1505          STORE G          ; store
        6842 1506 ;+
        6842 1507 ; Restore the following offsets.
        6842 1508 ; -
        6842 1509
0000001C 6842 1510 value_desc = 28
0000001C 6842 1511 str_len = 28
0000001E 6842 1512 dtype = 30
0000001F 6842 1513 class = 31
00000020 6842 1514 pointer = 32
00000024 6842 1515 data = 36
        6842 1516
        05 6842 1517          RSB          ; go continue loop
        6843 1518
        6843 1519 DEST_B_TO_H:
50 50 6CFD 6843 1520          CVTBH      R0, R0          ; convert
        2D 11 6847 1521          BRB      STORE_HFLOAT    ; go store
        6849 1522
        6849 1523 DEST_W_TO_H:
50 50 6DFD 6849 1524          CVTWH      R0, R0          ; convert
        27 11 684D 1525          BRB      STORE_HFLOAT
        684F 1526
        684F 1527 DEST_L_TO_H:
50 50 6EFD 684F 1528          CVTLH      R0, R0          ; convert
        21 11 6853 1529          BRB      STORE_HFLOAT    ; go store
        6855 1530
        6855 1531 DEST_F_TO_H:
50 50 98FD 6855 1532          CVTFH      R0, R0          ; convert
        1B 11 6859 1533          BRB      STORE_HFLOAT    ; go store
        685B 1534
        685B 1535 DEST_D_TO_H:
50 7E 50 70 685B 1536          MOVD      R0, -(SP)          ; save double
50 50 0C AD D0 685E 1537          MOVL      SF$L SAVE FP(FP), R0 ; pass FP to get scale
00000000 GF 16 6862 1538          JSB      G*BASS$SCALE_R1 ; get scale in R0 & R1
50 8E 50 67 6868 1539          DIVD3     R0, (SP)+, R0 ; descale for dest
50 50 50 32FD 686C 1540          CVTDH      R0, R0          ; convert to hfloat
        04 11 6870 1541          BRB      STORE_HFLOAT    ; go store
        6872 1542
        6872 1543 DEST_G_TO_H:
50 50 56FD 6872 1544          CVTGH      R0, R0          ; convert
        6876 1545          ; fall into store
        6876 1546
        6876 1547 STORE_HFLOAT:
50 54 5A D0 6876 1548          MOVL      R10, R4          ; pointer to dest descriptor
55 08 AE D0 6879 1549          MOVL      lower_bnd1+4(SP), R5 ; current row (extra longword
        687D 1550          ; on top of stack for jsb)
        56 5B D0 687D 1551          MOVL      R11, R6          ; current column
28 AE 50 70FD 6880 1552          MOVH      R0, DATA+4(SP)
        6885 1553 ;+
        6885 1554 ; Redefine the following offsets for the call to the STORE macro. The
        6885 1555 ; BSBW to here added 4 to the stack.
        6885 1556 ; -
```

```

00000020 6B85 1557
00000020 6B85 1558 value_desc = 32
00000020 6B85 1559 str_len = 32
00000022 6B85 1560 dtype = 34
00000023 6B85 1561 class = 35
00000024 6B85 1562 pointer = 36
00000028 6B85 1563 data = 40
        6B85 1564
        6B85 1565 STORE H ; store
        6C5A 1566 ;+
        6C5A 1567 ; Restore the following offsets.
        6C5A 1568 ; -
        6C5A 1569
0000001C 6C5A 1570 value_desc = 28
0000001C 6C5A 1571 str_len = 28
0000001E 6C5A 1572 dtype = 30
0000001F 6C5A 1573 class = 31
00000020 6C5A 1574 pointer = 32
00000024 6C5A 1575 data = 36
        6C5A 1576
05      6C5A 1577 RSB ; go continue loop
        6C5B 1578
        6C5B 1579 .END

```

BASSMAT SUB
Symbol Table

D 14

15-SEP-1984 23:52:49 VAX/VMS Macro V04-00
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1

Page 78
(6)

| | | | |
|-------------------|----------|----|----|
| BASS\$SCALE_R1 | ***** | X | 00 |
| BASS\$STOP | ***** | X | 00 |
| BASS\$FETCH_BFA | ***** | X | 00 |
| BASS\$FET_FA_B_R8 | ***** | X | 00 |
| BASS\$FET_FA_D_R8 | ***** | X | 00 |
| BASS\$FET_FA_F_R8 | ***** | X | 00 |
| BASS\$FET_FA_G_R8 | ***** | X | 00 |
| BASS\$FET_FA_H_R8 | ***** | X | 00 |
| BASS\$FET_FA_L_R8 | ***** | X | 00 |
| BASS\$FET_FA_W_R8 | ***** | X | 00 |
| BASS\$K_ARGDONMAT | ***** | X | 00 |
| BASS\$K_ARRMUSSAM | ***** | X | 00 |
| BASS\$K_DATTYPERR | ***** | X | 00 |
| BASS\$K_MATDIMERR | ***** | X | 00 |
| BASSMAT_REDIM | ***** | X | 00 |
| BASSMAT_SUB | 00000000 | RG | 02 |
| BASS\$STORE_BFA | ***** | X | 00 |
| BASS\$STO_FA_B_R8 | ***** | X | 00 |
| BASS\$STO_FA_D_R8 | ***** | X | 00 |
| BASS\$STO_FA_F_R8 | ***** | X | 00 |
| BASS\$STO_FA_G_R8 | ***** | X | 00 |
| BASS\$STO_FA_H_R8 | ***** | X | 00 |
| BASS\$STO_FA_L_R8 | ***** | X | 00 |
| BASS\$STO_FA_W_R8 | ***** | X | 00 |
| BYTE | 00000107 | R | 02 |
| BYTE_TO_BYTE | 00000139 | R | 02 |
| BYTE_TO_DOUBLE | 000008FA | R | 02 |
| BYTE_TO_FLOAT | 00000709 | R | 02 |
| BYTE_TO_GFLOAT | 00000AFE | R | 02 |
| BYTE_TO_HFLOAT | 00000CF8 | R | 02 |
| BYTE_TO_LONG | 00000518 | R | 02 |
| BYTE_TO_WORD | 00000327 | R | 02 |
| DEST_B_TO_D | 000068BC | R | 02 |
| DEST_B_TO_F | 000067AD | R | 02 |
| DEST_B_TO_G | 00006A2B | R | 02 |
| DEST_B_TO_H | 00006843 | R | 02 |
| DEST_B_TO_L | 0000669E | R | 02 |
| DEST_B_TO_W | 0000658F | R | 02 |
| DEST_CASE_B | 0000630D | R | 02 |
| DEST_CASE_D | 000063E1 | R | 02 |
| DEST_CASE_F | 000063AC | R | 02 |
| DEST_CASE_G | 00006416 | R | 02 |
| DEST_CASE_H | 0000644B | R | 02 |
| DEST_CASE_L | 00006377 | R | 02 |
| DEST_CASE_W | 00006342 | R | 02 |
| DEST_D_TO_B | 0000648F | R | 02 |
| DEST_D_TO_F | 000067BC | R | 02 |
| DEST_D_TO_G | 00006A43 | R | 02 |
| DEST_D_TO_H | 0000685B | R | 02 |
| DEST_D_TO_L | 000066AD | R | 02 |
| DEST_D_TO_W | 0000659E | R | 02 |
| DEST_F_TO_B | 0000648A | R | 02 |
| DEST_F_TO_D | 000068F2 | R | 02 |
| DEST_F_TO_G | 00006A3D | R | 02 |
| DEST_F_TO_H | 00006855 | R | 02 |
| DEST_F_TO_L | 000066A8 | R | 02 |
| DEST_F_TO_W | 00006599 | R | 02 |

| |
|------------------|
| DEST_G_TO_B |
| DEST_G_TO_D |
| DEST_G_TO_F |
| DEST_G_TO_H |
| DEST_G_TO_L |
| DEST_G_TO_W |
| DEST_H_TO_B |
| DEST_H_TO_D |
| DEST_H_TO_F |
| DEST_H_TO_G |
| DEST_H_TO_L |
| DEST_H_TO_W |
| DEST_L_TO_B |
| DEST_L_TO_D |
| DEST_L_TO_F |
| DEST_L_TO_G |
| DEST_L_TO_H |
| DEST_L_TO_W |
| DEST_MATRIX |
| DEST_W_TO_B |
| DEST_W_TO_D |
| DEST_W_TO_F |
| DEST_W_TO_G |
| DEST_W_TO_H |
| DEST_W_TO_L |
| DOUBLE |
| DOUBLE_TO_BYTE |
| DOUBLE_TO_DOUBL |
| DOUBLE_TO_FLOAT |
| DOUBLE_TO_GFLOA |
| DOUBLE_TO_HFLOA |
| DOUBLE_TO_LONG |
| DOUBLE_TO_WORD |
| DSC\$A_AO |
| DSC\$B_AFLAGS |
| DSC\$B_CLASS |
| DSC\$B_DIMCT |
| DSC\$B_DTYPE |
| DSC\$K_CLASS_BFA |
| DSC\$K_DTYPE_B |
| DSC\$K_DTYPE_D |
| DSC\$K_DTYPE_DSC |
| DSC\$K_DTYPE_G |
| DSC\$K_DTYPE_H |
| DSC\$L_L1_1 |
| DSC\$L_L1_2 |
| DSC\$L_L2_2 |
| DSC\$L_M1 |
| DSC\$L_M2 |
| DSC\$L_U1_1 |
| DSC\$L_U1_2 |
| DSC\$L_U2_2 |
| DSC\$V_FL_BOUNDS |
| DSC\$W_LENGTH |
| ERR_ARGDONMAT |
| ERR_ARRMUSSAM |
| ERR_DATTYPERR |

| | | |
|------------|---|----|
| 000064A5 | R | 02 |
| 0000690A | R | 02 |
| 000067D2 | R | 02 |
| 00006872 | R | 02 |
| 000066C3 | R | 02 |
| 000065B4 | R | 02 |
| 000064AB | R | 02 |
| 00006934 | R | 02 |
| 000067D8 | R | 02 |
| 00006A5A | R | 02 |
| 000066C9 | R | 02 |
| 000065BA | R | 02 |
| 00006485 | R | 02 |
| 000068E0 | R | 02 |
| 000067B7 | R | 02 |
| 00006A37 | R | 02 |
| 0000684F | R | 02 |
| 00006594 | R | 02 |
| = 0000000C | | |
| 00006480 | R | 02 |
| 000068CE | R | 02 |
| 000067B2 | R | 02 |
| 00006A31 | R | 02 |
| 00006849 | R | 02 |
| 000066A3 | R | 02 |
| 000038B3 | R | 02 |
| 000038E5 | R | 02 |
| 0000410D | R | 02 |
| 00003F03 | R | 02 |
| 000042FB | R | 02 |
| 000044F9 | R | 02 |
| 00003CF9 | R | 02 |
| 00003AEF | R | 02 |
| = 00000010 | | |
| = 0000000A | | |
| = 00000003 | | |
| = 0000000B | | |
| = 00000002 | | |
| = 000000BF | | |
| = 00000006 | | |
| = 0000000B | | |
| = 00000018 | | |
| = 0000001B | | |
| = 0000001C | | |
| = 00000018 | | |
| = 0000001C | | |
| = 00000024 | | |
| = 00000014 | | |
| = 00000018 | | |
| = 0000001C | | |
| = 00000020 | | |
| = 00000028 | | |
| = 00000007 | | |
| = 00000000 | | |
| 0000002A | R | 02 |
| 00000076 | R | 02 |
| 000000FA | R | 02 |

BASMAT SUB
Symbol Table

E 14

15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 79
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (6)

| | | | | | |
|-----------------|------------|----|----------------|------------|----|
| ERR_MATDIMERR | 00000069 R | 02 | LOOP_1ST_SUBGD | 00004F09 R | 02 |
| FLOAT | 00002AC8 R | 02 | LOOP_1ST_SUBGF | 00004D10 R | 02 |
| FLOAT_TO_BYTE | 00002AFA R | 02 | LOOP_1ST_SUBGG | 00005106 R | 02 |
| FLOAT_TO_DOUBLE | 000032BB R | 02 | LOOP_1ST_SUBGH | 00005302 R | 02 |
| FLOAT_TO_FLOAT | 000030CD R | 02 | LOOP_1ST_SUBGL | 00004B17 R | 02 |
| FLOAT_TO_GFLOAT | 000034BF R | 02 | LOOP_1ST_SUBGW | 0000491E R | 02 |
| FLOAT_TO_HFLOAT | 000036B9 R | 02 | LOOP_1ST_SUBHB | 00005534 R | 02 |
| FLOAT_TO_LONG | 00002EDC R | 02 | LOOP_1ST_SUBHD | 00005D18 R | 02 |
| FLOAT_TO_WORD | 00002CEB R | 02 | LOOP_1ST_SUBHF | 00005B1F R | 02 |
| GFLOAT | 000046F3 R | 02 | LOOP_1ST_SUBHG | 00005F11 R | 02 |
| GFLOAT_TO_BYTE | 00004725 R | 02 | LOOP_1ST_SUBHH | 00006111 R | 02 |
| GFLOAT_TO_DOUBL | 00004F09 R | 02 | LOOP_1ST_SUBHL | 00005926 R | 02 |
| GFLOAT_TO_GFLOA | 00004D10 R | 02 | LOOP_1ST_SUBHW | 0000572D R | 02 |
| GFLOAT_TO_HFLOA | 00005106 R | 02 | LOOP_1ST_SUBLB | 00001D0F R | 02 |
| GFLOAT_TO_LONG | 00005302 R | 02 | LOOP_1ST_SUBLD | 000024D0 R | 02 |
| GFLOAT_TO_WORD | 00004B17 R | 02 | LOOP_1ST_SUBLF | 000022DF R | 02 |
| HFLOAT | 0000491E R | 02 | LOOP_1ST_SUBLG | 000026D4 R | 02 |
| HFLOAT_TO_BYTE | 00005502 R | 02 | LOOP_1ST_SUBLH | 000028CE R | 02 |
| HFLOAT_TO_DOUBL | 00005534 R | 02 | LOOP_1ST_SUBLL | 000020F1 R | 02 |
| HFLOAT_TO_FLOAT | 00005D18 R | 02 | LOOP_1ST_SUBLW | 00001F00 R | 02 |
| HFLOAT_TO_GFLOA | 00005B1F R | 02 | LOOP_1ST_SUBWB | 00000F24 R | 02 |
| HFLOAT_TO_HFLOA | 00005F11 R | 02 | LOOP_1ST_SUBWD | 000016E5 R | 02 |
| HFLOAT_TO_LONG | 00006111 R | 02 | LOOP_1ST_SUBWF | 000014F4 R | 02 |
| HFLOAT_TO_WORD | 00005926 R | 02 | LOOP_1ST_SUBWG | 000018E9 R | 02 |
| INIT_ONE_SUB | 0000572D R | 02 | LOOP_1ST_SUBWH | 00001AE3 R | 02 |
| INIT_TWO_SUBS | 00000037 R | 02 | LOOP_1ST_SUBWL | 00001303 R | 02 |
| LONG | 00000083 R | 02 | LOOP_1ST_SUBWW | 00001115 R | 02 |
| LONG_TO_BYTE | 00001CDD R | 02 | LOOP_2ND_SUBBB | 0000013C R | 02 |
| LONG_TO_DOUBLE | 00001D0F R | 02 | LOOP_2ND_SUBBD | 000008FD R | 02 |
| LONG_TO_FLOAT | 000024D0 R | 02 | LOOP_2ND_SUBBF | 0000070C R | 02 |
| LONG_TO_GFLOAT | 000022DF R | 02 | LOOP_2ND_SUBBG | 00000B01 R | 02 |
| LONG_TO_HFLOAT | 000026D4 R | 02 | LOOP_2ND_SUBBH | 00000CFB R | 02 |
| LONG_TO_LONG | 000028CE R | 02 | LOOP_2ND_SUBBL | 0000051B R | 02 |
| LONG_TO_WORD | 000020F1 R | 02 | LOOP_2ND_SUBBW | 0000032A R | 02 |
| LOOP_1ST_SUBBB | 00001F00 R | 02 | LOOP_2ND_SUBDB | 000038E8 R | 02 |
| LOOP_1ST_SUBBD | 00000139 R | 02 | LOOP_2ND_SUBDD | 00004110 R | 02 |
| LOOP_1ST_SUBBF | 000008FA R | 02 | LOOP_2ND_SUBDF | 00003F06 R | 02 |
| LOOP_1ST_SUBBG | 00000709 R | 02 | LOOP_2ND_SUBDG | 000042FE R | 02 |
| LOOP_1ST_SUBBH | 00000AFE R | 02 | LOOP_2ND_SUBDH | 000044FC R | 02 |
| LOOP_1ST_SBBH | 00000CF8 R | 02 | LOOP_2ND_SBDL | 00003CFC R | 02 |
| LOOP_1ST_SBBL | 00000518 R | 02 | LOOP_2ND_SBDW | 00003AF2 R | 02 |
| LOOP_1ST_SBBW | 00000327 R | 02 | LOOP_2ND_SBF8 | 00002AFD R | 02 |
| LOOP_1ST_SBD8 | 000038E5 R | 02 | LOOP_2ND_SBF9 | 000032BE R | 02 |
| LOOP_1ST_SBD9 | 0000410D R | 02 | LOOP_2ND_SBFF | 000030D0 R | 02 |
| LOOP_1ST_SBDG | 00003F03 R | 02 | LOOP_2ND_SBFG | 000034C2 R | 02 |
| LOOP_1ST_SBDH | 000042FB R | 02 | LOOP_2ND_SBFH | 000036BC R | 02 |
| LOOP_1ST_SBDL | 000044F9 R | 02 | LOOP_2ND_SBFJ | 00002EDF R | 02 |
| LOOP_1ST_SBDW | 00003CF9 R | 02 | LOOP_2ND_SBFK | 00002CEE R | 02 |
| LOOP_1ST_SBF8 | 00003AEF R | 02 | LOOP_2ND_SBFM | 00004728 R | 02 |
| LOOP_1ST_SBF9 | 00002AFA R | 02 | LOOP_2ND_SBGD | 00004F0C R | 02 |
| LOOP_1ST_SBFJ | 000032BB R | 02 | LOOP_2ND_SBGF | 00004D13 R | 02 |
| LOOP_1ST_SBFK | 000030CD R | 02 | LOOP_2ND_SBGG | 00005109 R | 02 |
| LOOP_1ST_SBFM | 000034BF R | 02 | LOOP_2ND_SBGH | 00005305 R | 02 |
| LOOP_1ST_SBFN | 000036B9 R | 02 | LOOP_2ND_SBGL | 00004B1A R | 02 |
| LOOP_1ST_SBFJ | 00002EDC R | 02 | LOOP_2ND_SBGW | 00004921 R | 02 |
| LOOP_1ST_SBFK | 00002CEB R | 02 | LOOP_2ND_SBHB | 00005537 R | 02 |
| LOOP_1ST_SBG8 | 00004725 R | 02 | LOOP_2ND_SBHD | 00005D1B R | 02 |

BASSMAT SUB
Symbol Table

F 14

15-SEP-1984 23:52:49 VAX/VMS Macro V04-00 Page 80
6-SEP-1984 10:31:08 [BASRTL.SRC]BASMATSUB.MAR;1 (6)

| | | | |
|-----------------|------------|---|----|
| LOOP_2ND_SUBHF | 00005B22 | R | 02 |
| LOOP_2ND_SUBHG | 00005F14 | R | 02 |
| LOOP_2ND_SUBHH | 00006114 | R | 02 |
| LOOP_2ND_SUBHL | 00005929 | R | 02 |
| LOOP_2ND_SUBHW | 00005730 | R | 02 |
| LOOP_2ND_SUBLB | 00001D12 | R | 02 |
| LOOP_2ND_SUBLD | 000024D3 | R | 02 |
| LOOP_2ND_SUBLF | 000022E2 | R | 02 |
| LOOP_2ND_SUBLG | 000026D7 | R | 02 |
| LOOP_2ND_SUBLH | 000028D1 | R | 02 |
| LOOP_2ND_SUBLL | 000020F4 | R | 02 |
| LOOP_2ND_SUBLW | 00001F03 | R | 02 |
| LOOP_2ND_SUBWB | 00000F27 | R | 02 |
| LOOP_2ND_SUBWD | 000016E8 | R | 02 |
| LOOP_2ND_SUBWF | 000014F7 | R | 02 |
| LOOP_2ND_SUBWG | 000018EC | R | 02 |
| LOOP_2ND_SUBWH | 00001AE6 | R | 02 |
| LOOP_2ND_SUBWL | 00001306 | R | 02 |
| LOOP_2ND_SUBWW | 00001118 | R | 02 |
| LOWER_BND1 | = 00000004 | | |
| LOWER_BND2 | = 00000000 | | |
| MTH\$DINT R4 | ***** | X | 00 |
| SAVE_SRC2 | = 0000000C | | |
| SEPARATE_DTYPES | 000000CB | R | 02 |
| SF\$L_SAVE_FP | = 0000000C | | |
| SRC1_MATRIX | = 00000004 | | |
| SRC2_MATRIX | = 00000008 | | |
| STORE_BYTE | 000064AF | R | 02 |
| STORE_DOUBLE | 0000694B | R | 02 |
| STORE_FLOAT | 000067DC | R | 02 |
| STORE_GFLOAT | 00006A5E | R | 02 |
| STORE_HFLOAT | 00006B76 | R | 02 |
| STORE_LONG | 000066CD | R | 02 |
| STORE_WORD | 000065BE | R | 02 |
| UPPER_BND1 | = 00000008 | | |
| WORD | 00000EF2 | R | 02 |
| WORD_TO_BYTE | 00000F24 | R | 02 |
| WORD_TO_DOUBLE | 000016E5 | R | 02 |
| WORD_TO_FLOAT | 000014F4 | R | 02 |
| WORD_TO_GFLOAT | 000018E9 | R | 02 |
| WORD_TO_HFLOAT | 00001AE3 | R | 02 |
| WORD_TO_LONG | 00001303 | R | 02 |
| WORD_TO_WORD | 00001115 | R | 02 |

! Psect synopsis !

| PSECT name | Allocation | PSECT No. | Attributes | | | | | | | | | | | | | | |
|------------|-------------------|-----------|------------|-----|-----|-----|-----|-------|-------|------|-------|-------|------|--|--|--|--|
| ABS | 00000000 (0.) | 00 (0.) | NOPIC | USR | CON | ABS | LCL | NOSHR | NOEXE | NORD | NOWRT | NOVEC | BYTE | | | | |
| \$ABSS | 00000000 (0.) | 01 (1.) | NOPIC | USR | CON | ABS | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE | | | | |
| _BASSCODE | 00006C5B (27739.) | 02 (2.) | PIC | USR | CON | REL | LCL | SHR | EXE | RD | NOWRT | NOVEC | LONG | | | | |

-----+
! Performance indicators !
-----+

| Phase | Page faults | CPU Time | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| ----- | ----- | ----- | ----- |
| Initialization | 32 | 00:00:00.09 | 00:00:00.42 |
| Command processing | 131 | 00:00:00.64 | 00:00:03.61 |
| Pass 1 | 1084 | 00:00:42.00 | 00:01:32.71 |
| Symbol table sort | 7 | 00:00:02.22 | 00:00:05.61 |
| Pass 2 | 1114 | 00:00:10.96 | 00:00:24.88 |
| Symbol table output | 33 | 00:00:00.26 | 00:00:00.82 |
| Psect synopsis output | 4 | 00:00:00.03 | 00:00:00.11 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 2408 | 00:00:56.21 | 00:02:08.17 |

The working set limit was 900 pages.
320586 bytes (627 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 421 non-local and 902 local symbols.
1579 source lines were read in Pass 1, producing 84 object records in Pass 2.
37 pages of virtual memory were used to define 11 macros.

-----+
! Macro library statistics !
-----+

| Macro library name | Macros defined |
|--|----------------|
| ----- | ----- |
| _\$255\$DUA28:[BASRTL.OBJ]BASRTL.MLB;1 | 2 |
| -\$255\$DUA28:[SYSLIB]STARLET.MLB;2 | 5 |
| TOTALS (all libraries) | 7 |

493 GETS were required to define 7 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:BASMATSUB/OBJ=OBJ\$:BASMATSUB MSRC\$:BASMATSUB/UPDATE=(ENH\$:BASMATSUB)+LI

0027 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BASMATSUB
LIS

BASMATSCA
LIS

BASMATRN
LIS